

10005.1006

UNITED STATES PATENT APPLICATION

FOR

DATABASE SIZING AND  
DIAGNOSTIC UTILITY

INVENTOR:

VINCENT CIVETTA  
INNA BROVMAN  
STEVE FABIAN  
ISABEL ESPINA

PREPARED BY:

THE HECKER LAW GROUP  
1925 Century Park East  
Suite 2300  
Los Angeles, CA 90067

(310) 286-0377

~~CERTIFICATE OF MAILING~~

~~This is to certify that this correspondence is being deposited  
with the United States Postal Service with sufficient postage as  
Express Mail Label No. EL557989242US  
in an envelope addressed to: Assistant Commissioner for  
Patents Washington, D.C. 20231 on:~~

~~FEBRUARY 25 2000~~

~~Signature Elaine Wells Date 2/25/00~~

CERTIFICATE OF MAILING

This is to certify that this correspondence is being deposited  
with the United States Postal Service with sufficient postage as  
Express Mail No. EL938710123US in  
an envelope addressed to: Mail Stop Patent Application  
Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313  
on: August 26, 2003

Signature Mario Federis Date 08/26/2003

This application claims the benefit of U.S. Provisional Application No. \_\_\_\_\_, filed on February 26, 1999, entitled "Sizing and Diagnostic Utility," the specification of which is herein incorporated by reference.

5           Portions of the disclosure of this patent document contain material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office file or records, but otherwise reserves all copyright rights whatsoever.

10

## BACKGROUND OF THE INVENTION

### 1.   FIELD OF THE INVENTION

15

This invention relates to the field of databases.

### 2.   BACKGROUND ART

20           Installing and maintaining a database is a complex and time consuming task. Typically, a specially trained and/or certified person or team is required for installing and setting up a database. Maintaining the database during operation often requires that a service team be contacted to provide support.

Another problem associated with databases is that the database and the application using the database are often independently designed and configured, leading to fragmentation and decreased performance. Further, over time, the data residing in the database changes, as well as the relationships between the data. This too causes fragmentation, even in databases that may have been well-  
5 configured initially to suit the original data needs of the user.

Some databases, such as the Oracle™ database, are organized into “tablespaces.” Tablespaces are physical allocations of space that hold related  
10 objects such as tables or indexes. Tables and indexes are created in specific tablespaces. These tables and indexes are created with an initial allocation within a tablespace, which is referred to as an “extent.” If a table or index runs out of space in the initial extent, a further pre-defined extent may be allocated. New extents are often allocated from contiguous free space within a tablespace. As a  
15 tablespace becomes fragmented, the tablespace’s free space can be left in such small blocks that the free space is virtually unusable. Also, when tables or indexes have too many extents, the database’s performance degrades. Multiple extents require more physical I/O operations to accomplish a query.

20 A database solution is desired that minimizes the need for specially trained personnel for configuring and maintaining a database, and addresses the problems associated with database fragmentation, both initially and over time.

## SUMMARY OF THE INVENTION

The invention is a system for automated installation and maintenance of databases. One or more embodiments provide a user interface (or wizard) that  
5 obtains information from a user regarding aspects of the network environment and application data requirements. Using the information obtained from the user, a sizing process builds a database, or resizes an existing database, to efficiently match the needs of the user. An automated maintenance process self monitors, diagnoses, and fixes database problems, such as by rebuilding table  
10 keys and indexes. When the diagnostic cannot fix a problem, appropriate notification takes place.

In one embodiment, the user information is processed using sizing formulas to obtain values for building the database. Database scripts and  
15 command files are generated which, when executed, build the appropriately configured database. Also, in accordance with the user information, scripts and command files may be generated that will implement a database backup process upon a user-specified schedule.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a general-purpose computer upon which an embodiment of the invention may be implemented.

5

Figure 2 is a block diagram of an embodiment of the invention.

Figure 3 is a flow diagram of a sizing process in accordance with an embodiment of the invention.

10

Figure 4 is a flow diagram of a maintenance process in accordance with an embodiment of the invention.

Figures 5A-5C are flow diagrams of steps within the process of Figure 4.

## DETAILED DESCRIPTION OF THE INVENTION

In the following description, numerous specific details are set forth to provide a more thorough description of embodiments of the invention. It will be apparent, however, to one skilled in the art, that the invention may be practiced without these specific details. In other instances, well known features have not been described in detail so as not to obscure the invention.

### Embodiment of General-Purpose Computer Environment

10

An embodiment of the invention can be implemented as computer software in the form of computer readable program code executed on a general-purpose computer such as computer 100 illustrated in Figure 1. A keyboard 110 and mouse 111 are coupled to a bi-directional system bus 118. The keyboard and mouse are for introducing user input to the computer system and communicating that user input to central processing unit (CPU) 113. Other suitable input devices may be used in addition to, or in place of, the mouse 111 and keyboard 110. I/O (input/output) unit 119 coupled to bi-directional system bus 118 represents such I/O elements as a printer, A/V (audio/video) I/O, etc.

20

Computer 100 includes a video memory 114, main memory 115 and mass storage 112, all coupled to bi-directional system bus 118 along with keyboard 110, mouse 111 and CPU 113. The mass storage 112 may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology. Bus 118 may contain, for

example, thirty-two address lines for addressing video memory 114 or main memory 115. The system bus 118 also includes, for example, a 32-bit data bus for transferring data between and among the components, such as CPU 113, main memory 115, video memory 114 and mass storage 112. Alternatively,  
5 multiplex data/address lines may be used instead of separate data and address lines.

In one embodiment of the invention, the CPU 113 is a microprocessor manufactured by Motorola, such as the 680X0 processor or a microprocessor  
10 manufactured by Intel, such as the 80X86, or Pentium processor, or a SPARC microprocessor from Sun Microsystems. However, any other suitable microprocessor or microcomputer may be utilized. Main memory 115 is comprised of dynamic random access memory (DRAM). Video memory 114 is a dual-ported video random access memory. One port of the video memory 114  
15 is coupled to video amplifier 116. The video amplifier 116 is used to drive the cathode ray tube (CRT) raster monitor 117. Video amplifier 116 is well known in the art and may be implemented by any suitable apparatus. This circuitry converts pixel data stored in video memory 114 to a raster signal suitable for use by monitor 117. Monitor 117 is a type of monitor suitable for displaying graphic  
20 images.

Computer 100 may also include a communication interface 120 coupled to bus 118. Communication interface 120 provides a two-way data communication coupling via a network link 121 to a local network 122. For example, if  
25 communication interface 120 is an integrated services digital network (ISDN)

card or a modem, communication interface 120 provides a data communication connection to the corresponding type of telephone line, which comprises part of network link 121. If communication interface 120 is a local area network (LAN) card, communication interface 120 provides a data communication connection  
5 via network link 121 to a compatible LAN. Wireless links are also possible. In any such implementation, communication interface 120 sends and receives electrical, electromagnetic or optical signals which carry digital data streams representing various types of information.

10 Network link 121 typically provides data communication through one or more networks to other data devices. For example, network link 121 may provide a connection through local network 122 to host computer 123 or to data equipment operated by an Internet Service Provider (ISP) 124. ISP 124 in turn provides data communication services through the world wide packet data  
15 communication network now commonly referred to as the "Internet" 125. Local network 122 and Internet 125 both use electrical, electromagnetic or optical signals which carry digital data streams. The signals through the various networks and the signals on network link 121 and through communication interface 120, which carry the digital data to and from computer 100, are  
20 exemplary forms of carrier waves transporting the information.

Computer 100 can send messages and receive data, including program code, through the network(s), network link 121, and communication interface 120. In the Internet example, server 126 might transmit a requested code for an

application program through Internet 125, ISP 124, local network 122 and communication interface 120.

5 The received code may be executed by CPU 113 as it is received, and/or stored in mass storage 112, or other non-volatile storage for later execution. In this manner, computer 100 may obtain application code in the form of a carrier wave.

10 The computer systems described above are for purposes of example only. An embodiment of the invention may be implemented in any type of computer system or programming or processing environment.

#### Embodiment of Database Sizing and Diagnostic Utility

15 Embodiments of the invention are directed at building and maintaining a database in which the sizing allocations conform to the needs of the user application that is using the database. The initial configuration of the database is performed based on user-provided information about the networking environment and assumptions about the application needs of the user. The user  
20 assumptions may become less accurate over time, in which case, an embodiment of the invention may be used to obtain new assumptions from the user regarding application needs. Those new assumptions are then used to resize the database.

As an example, an Oracle database may be used to implement a payroll system application. In such a case, user information is obtained in the form of assumptions about the projected number of employees in the company, the number and types of payroll items that apply to the average employee, etc. The database sizing and diagnostic utility is configured with formulas for converting those payroll assumptions into table parameters that are then used to size the database.

An embodiment of the invention is illustrated in Figure 2. As shown, a database sizing and diagnostic utility 200 comprises a database building/sizing process 201 and a database maintenance/diagnostic process 204. Within database building/sizing process 201 are a graphic user interface (GUI) 202 (also referred to herein as a “wizard”) and index/table sizing formulas 203.

In one embodiment, GUI 202 presents a sequence of panels for receiving user input. It will be obvious, however, that the invention is not limited to those GUI mechanisms, and that any form of user interface may be employed (e.g., an audio interface). GUI 202 is used to ask questions of the user and to obtain user information in return. The user information comprises information about the networking environment, assumptions about the application-specific needs of the user, and user preferences for database backup operations.

The index/table sizing formulas 203 are used to transform the user information into database sizing parameters that are incorporated into database

scripts and command files 205 for building and sizing (or resizing) the database 207. Backup scripts and command files 206 are generated by database building and sizing process 201 from the user-specified backup preferences.

- 5           Database maintenance/diagnostic process 204 executes on a periodic basis to evaluate the performance of the database (though a user may also manually prompt the database maintenance/diagnostic process 204 to execute). Entries made to a logfile may serve as an indicator to a user that it may be appropriate to resize the database 207. Problems with tables and indexes which are identified  
10 by the database maintenance/diagnostic process 204 are automatically fixed when possible.

#### Database Building/Sizing Process

- 15           The database building and sizing process 203 is used by the user to optionally install and configure the database engine on their network server, and to build a pre-sized database for a given database application. The advantage of presizing the database correctly is a reduction in tablespace fragmentation and increased performance. Presizing the database, along with the automated  
20 database maintenance/diagnostic process 204, permit a user to install a database application without requiring an on-site certified database specialist to manage the database.

Figure 3 is a flow diagram of the database building/sizing process 201 in accordance with an embodiment of the invention. In step 300, process 201 optionally installs and configures the database engine on the user's server machine. If this is a resizing operation or if the database engine is already installed, step 300 is skipped. In step 301, the database building/sizing process 201 collects information from the user via GUI 202 (e.g., in interview format).

Step 301 is subdivided into component steps 301A-301B. In step 301A, the user information obtained includes information regarding the user's network environment (number of users and amount of RAM, for instance). In step 301B, process 201 obtains information from the user regarding how many drives the user wants the database to span. In step 301C, the user information obtained concerns the data requirements of the database application, e.g., for a payroll application, the user's payroll data requirements (number of employees, number of company codes, and amount of history to keep online, for instance). In step 301D, GUI 202 obtains the user's preferences for database backup operations, including the backup mode (if more than one mode is available) and the backup schedule.

In step 302, the database building/sizing process 201 generates a series of instructions, for example SQL scripts and Windows NT command files, in accordance with the user information obtained in step 301. Specifically, in step 302A, instructions are generated to physically create a database that will sufficiently house the user's data, and that will be optimized and tuned to

perform as well as possible, e.g., based on the network environment information and other user information. In step 302B, instructions are generated to implement the specified periodic backup operation. In step 303, database building/sizing process 201 executes the command files to physically build the  
5 database.

In one embodiment of the invention, database building/sizing process 201 and its constituent GUI 202 are implemented as a "wizard" application. The user is presented with a sequence of panels from which the user information of step  
10 301 is obtained. One possible implementation of such a wizard application is described in Appendix A, with corresponding pseudo-code, under the heading "dbsizer.exe: Oracle Sizing Wizard." A database utility program for performing certain database procedures with command line parameters is described in Appendix A under the heading of "brunner.exe: Database Utility Program," with  
15 accompanying pseudo-code and source code.

#### Database Maintenance/Diagnostic Process

The database maintenance/diagnostic process 204 is an unattended  
20 database diagnostic and auto-maintenance utility used by the user to perform the following database procedures:

1. check the database for tablespace fragmentation
2. check the tablespaces for available free space
- 25 3. check the hard drives for available free space

4. fix any problems that can be fixed automatically without risk

The database maintenance/diagnostic process 204 is scheduled to run at intervals, e.g., once per week, and terminates automatically upon completion.

- 5 Process messages and errors are written to a logfile for user reference.

The general flow of the maintenance/diagnostic process is illustrated in Figure 4. In step 401, all objects (e.g., tables and indexes) are analyzed, and information is gathered regarding those objects that can be fixed automatically and those objects that require manual fixing. In step 402, the database performance is evaluated, with problem areas noted in the logfile. In step 403, those tables that were designated for automatic fixing in step 401 are fixed. In step 404, indexes are rebuilt where necessary. Steps 401-403 are described in more detail below with reference to Figures 5A-5C, respectively.

15

Figure 5A is directed to table analysis and the gathering of information about the database. In step 500, the database maintenance/diagnostic process 204 coalesces all tablespaces, and, in step 501, builds a list of all high-risk objects with extents greater than one. Objects are considered high-risk if their extents are numerous enough that an automatic fixing operation could compromise their integrity. These high-risk objects are listed in the logfile, in step 502, as objects that will require manual fixing. In step 503, a report is generated on the database internals. In step 504, all tables are analyzed, and in step 505, a list is made of those objects that should be automatically fixed by the database maintenance/diagnostic process.

25

Figure 5B illustrates steps for performing database performance analysis. In step 506, a table is generated that contains entries for database performance values in different categories. In step 507, performance criteria are obtained that specify, for example, error levels and warning levels for each performance category. Step 508, comprising steps 508A-508D, is performed for each entry in the performance table generated in step 506. In step 508A, the performance value for one entry in the table is compared with the corresponding error level. If the performance value is above the specified error level, an error message is written to the logfile in step 508B, and the process continues at step 509. If, in step 508A, the performance value is not above the error level, then the performance value is compared with the warning level in step 508C. If the performance value is above the error level, a warning message is written to the logfile in step 508D before proceeding to step 509. If the performance value is not above the warning level in step 508C, the process continues at step 509.

Step 509, comprising steps 509A-509B, is performed for each hard drive upon which the database is spread. In step 509A, the free space of the hard drive is compared with a minimum space threshold value needed to support the database. If the free space available does not meet the minimum space threshold value, a warning message is written to the logfile in step 509B.

Figure 5C illustrates one method for fixing tables in accordance with an embodiment of the invention. In step 510, the database maintenance/diagnostic process 204 opens the list of tables that can be automatically fixed. In step 511,

the first table listed is selected for fixing. In step 512, a DDL script is generated that will rebuild the primary keys of the table, and, in step 513, a DDL script is similarly generated to rebuild the table's foreign keys.

- 5           In step 514, the table data is exported to an export file and, in step 515, the table is dropped. In step 516, the table data in the export file is imported back in. In steps 517 and 518, respectively, the primary key and foreign key rebuild scripts are run to fix the table. In step 519, if the current table is not the last table on the list, the next table is selected and the process continues at step 512;
- 10   otherwise, the process continues in step 404 of Figure 4.

One possible implementation of database maintenance/diagnostic process 204 is described in Appendix A, with corresponding pseudo-code and source code, under the heading "hwb.exe: Health and Well-Being Utility."

15

Thus, a database sizing and diagnostic utility has been described in conjunction with one or more embodiments. The invention is defined by the claims and their full scope of equivalents.

## dbSizer.exe

### Oracle Sizing Wizard

#### Overview

The **dbSizer** utility is used by the client to (optionally) install and configure the Oracle Database engine on their Network Server, and to build a pre-sized ADP PC/Payroll for Windows database. The advantage of pre-sizing the database correctly is a reduction in tablespace fragmentation and increased performance. This process of pre-sizing the database along with the Health-and-Well Being utility (hwb.exe) allows ADP to install an Oracle based application without requiring an Oracle DBA on-site to manage the database.

#### Process Overview

The Oracle Sizing Wizard ('the wizard') collects information from the user regarding their network environment (# users, amount of RAM, etc), their payroll data requirements (# of employees, # of company codes, amount of history to keep online, etc) and generates a series of SQL scripts and NT command files to physically create a database that will sufficiently house the client's data and perform as well as possible. The steps break down as follows:

1. Install and Configure Oracle on the client's Server (if requested, this is an optional step).
2. Gather information about the user's network environment.
3. Determine how many drives the user want to spread the Oracle database over (the more the better).
4. Gather information about the client's company and their payroll data requirements.
5. Ask the user which backup method they would like to use to backup their PCPW database (The wizard can install three different types of automated backups, as well as support a custom one supplied by the client)
6. Ask the user when they would like the backup to take place (schedule)
7. Build the scripts and command files to build the database sized according to the user's input, and build script and command files to implement the backup method chosen by the user.
8. Execute the command files to physically build the database,

### Architectural Overview

The wizard is a Visual Basic 5.0 application that looks like a standard wizard. It appears to be one window that asks a series of questions and performs a task at the end when all necessary information has been gathered. It can be thought of as a 'interview-style' application.

Technically, each panel is a separate window and as the user presses the Back or Next button, to display the previous or next panel, the application hides the current window and displays the next one.

Control information is stored in an Access97 format database named **default.mdb** There are a number of tables in this database that are used by the wizard.

Table Name	Description / Usage
DBMisc	Miscellaneous information. Backup Method and Schedule
DBOptions	Items that are used to create the INITPCPW.ORA file. These items control the configuration of the Oracle database engine
ExistingTablespaces	Tablespaces and current size. Used by the wizard in resize mode to resize existing tablespaces.
FileLocations	Location of Oracle components and the PCPW admin folder
Indexes	Index sizing formulas and which tablespace each index is associated with
OracleComponents	For each Oracle Version supported, which components are to be installed by the automatic response script.
OracleVersions	Supported Oracle Versions
RangedObjects	Ranged formulas. These formulas override the formulas in Indexes, Tables and DBOptions. They allow multiple formulas to be defined for different ranges of NUMBER OF EMPLOYEES
Tables	Table sizing formulas and which tablespace each table is associated with
Tablespace	List of tablespaces

VariablesNNNNNN

There is one table for each Language supported. The NNNNNN value must match the Language id stored in the OS registry. These tables contain the prompts for Network Enviroment questions and Company Information questions.

## Pseduo-Code

### **' Panel 1 –The Welcome panel (frmPage1)**

get the OS language from the registry  
initialize program variables and counters  
search for the ADPSETUP.INI file  
    for each addressable drive  
        look in VAD\PCPW.DSK\DISK1  
    if not found  
        for each addressable drive  
            search all folders for ADPSETUP.INI  
    end if  
if not found  
    display error message  
    exit  
end if  
retrieve the Server's IP address from the ADPSETUP.INI file  
retrieve the location of the Migrate folder from the ADPSETUP.INI file  
*' Navigation*  
*' Back is always disabled*  
*' Next takes you to Panel 2 – Install Oracle (frmPage2)*

### **' Panel 2 –Install Oracle (frmPage2)**

open the default database (default.mdb)  
if it's not found in the current folder  
    pop a dialog so the user can tell you where it is.  
end if  
If we're running in Design mode  
    Display the **Load Configuration** push button  
end if  
*' Navigation*  
*' Back takes you to Panel 1 – Welcome (frmPage1)*  
*' Next has the following processing logic*  
    if the user wants the wizard to install Oracle  
        if Oracle is RUNNING (check for active service)  
            display error message  
            exit  
        end if  
        pop a dialog box to get the Server IP address (default from  
ADPSETUP.INI)  
        If the user changed the IP address  
            Write the new value to the ADPSETUP.INI file  
        End if  
        Search for the Oracle CD

```

        Run the Oracle installation program with a scripted response file
        If it fails
            Exit
        endif
    End if
    Search for an existing PCPW database
    If found
        Ask the user if they want to resize the existing database
        If they say no
            Exit
        End if
        If they say yes
            Make sure the instance is running and the database is up
        End if
    End if
    If we installed Oracle
        Display Panel 4 – Network Environment (frmNetworkEnv)
    Else
        Display Panel 3 – Where is Oracle (frmPage3)
    End if

‘ Panel 3 – Where is Oracle (frmPage3)
retrieve the default locations for the Oracle files
‘ Navigation
‘ Back takes you to Panel 2 – Install Oracle (frmPage2)
‘ Next has the following processing logic
    if we’re not in development mode
        verify the locations entered by the user
            BIN should contain ORADIM73.EXE
            RDBMS should contain CATALOG.SQL
    End if
    Make sure the version of Oracle is 7.3.4...
    Save the new locations as the defaults
    If we’re in RESIZE mode
        Display Panel 6 – Company Information (frmPage5)
    Else
        Display Panel 4 – Network Environment (frmNetworkEnv)
    End if

‘ Panel 4 – Network Environment (frmNetworkEnv)
load all Network questions from the database into the grid
‘ Navigation
‘ Back has the following processing logic
    if the wizard installed Oracle
        Display Panel 2 – Install Oracle (frmPage2)
    Else
        Display Panel 3 – Where is Oracle (frmPage3)
    end if
‘ Next has the following processing logic
    if we’re in DEVELOPMENT mode
        Display Database Options (frmPage4)
        ‘ NOTE: This is a DEVELOPMENT mode ONLY panel
    Else
        Display Panel 5 – Drives (frmDrives)
    End if

```

```

end if

' Panel 5 – Drives (frmDrives)
load combo boxes
  for each addressable drive
    make sure we can write to it
    if we can
      determine amount of free space
      add it to all 9 list boxes
    end if
  next drive
sort all 9 list boxes by free space
add <None> item to the top of each list box
for each list box
  select the drive with the most space free that hasn't been selected yet
next
' Navigation
' Back has the following processing logic
  if we're in DEVELOPMENT mode
    Display Database Options (frmPage4)
    ' NOTE: This is a DEVELOPMENT mode ONLY panel
  Else
    Display Panel 4 – Network Environment (frmNetworkEnv)
  end if
' Next has the following processing logic
  Display Panel 6 – Company Information (frmPage5)

' Panel 6 – Company Information (frmPage5)
load all company questions from the database into the grid
' Navigation
' Back has the following processing logic
  if we're in RESIZE mode
    if the wizard installed Oracle
      Display Panel 2 – Install Oracle (frmPage2)
    Else
      Display Panel 3 – Where is Oracle (frmPage3)
    end if
  else
    Display Panel
  end if
' Next has the following processing logic
  if we're in RESIZE mode
    Display Panel 9b – Resize (frmPage9)
  Else
    Display Panel 7 – Backup Information (frmPage6)
  end if

' Panel 7 – Backup Information (frmPage6)
display editable form with current values from default.mdb
' Navigation
' Back has the following processing logic
  Display Panel 6 – Company Information (frmPage5)
' Next has the following processing logic

```

## Display Panel 8 – Backup Schedule (frmPage7)

### **' Panel 8 – Backup Schedul (frmPage7)**

display editable form with current values from default.mdb

if we're in DEVELOPMENT mode

display the "Save Configuration" push button

end if

*' Navigation*

*' Back has the following processing logic*

Display Panel 7 – Backup Information (frmPage6)

*' Next has the following processing logic*

Based upon the number of drives selected

Set the target drive for each database element

*' See the functional spec for more information*

Display Panel 9a – Please wait, Database Creation Scripts (frmPage8)

### **' Panel 9a – Please wait . Database Creation Scripts (frmPage8)**

*' Create the scripts and command files to build the database*

*' a progress bar is displayed during this panel*

store all the user id's and encoded passwords in the NT Server's registry

make sure all necessary folders exist

if any do not

create them

end if

make sure there's at least 1 MEG free for scripts on the 1<sup>st</sup> drive

create the scripts and command files

create the INITPCPW.ORA file

create the SETUPDB.SQL file

create the TABPCPW.SQL file

take the TABXXX.TML file

merge the table sizing formulas from default.mdb

create the IDXPCPW.SQL file

take the IDXXXX.TML file

merge the index sizing formulas from default.mdb

create the backup scripts and command files

create the AT schedule entry

copy all required files from the DBSIZER folder to the ADMIN folder

create the command files to create the database

backup the PCPW registry entries to a PCPW.REG file in the ADMIN folder

*' Navigation*

*' the user has no choice, as soon as all files are created*

Display Panel 10 – Next Steps (frmNextSteps)

### **' Panel 9b – Please wait, Database Resize Scripts (frmPage9)**

*' Create the scripts and command files to resize the database*

*' a progress bar is displayed during this panel*

create the scripts and command files

calculate size needed for each table

calculate size needed for each index

rollup the sizes into the tablespaces

for each tablespace

determine the current size

compare it to the new size

if the new size is bigger

```

        calculate the difference
        find a drive which can handle the difference
            check the drive the current tablespace is on
            if it fits, use it
            if not
                check other drives that are host PCPW data
                if it fits and passes the neighbor rules
                    ' Neighbor rule state which tablespaces
can
                    ' live on the same drives as others
                    ' see the functional spec for more
information
                use it
            end if
            if we found a new home,
                build a script to create a new datafile for the
tablespace
            else
                pop a dialog and ask the user for a new drive
                if they give one
                    make sure it has enough room
                    if so
                        build the script
                    else
                        exit
                    end if
                end if
            end if
        end if
    end if
next tablespace
' Navigation
' the user has no choice, as soon as all files are created
Display Panel 10 – Next Steps (frmNextSteps)

' Panel 10 – Next Steps (frmNextSteps)
display a summary of the size of the database to be created or resized
' Navigation
' Create Database button pushed
If in RESIZE mode
    Display Panel 11b – Database Resize in Process (frmResize)
else
    Display Panel 11a – Database Creation in Process (frmCreation)
' Cancel
warn the user that if they cancel, they have to start over
if they say okay
    delete scripts and command files
    exit
end if

' Panel 11a – Database Creation in Process (frmCreation)
Display a checklist of steps to create the database
Execute the command file createdb.cmd
As each step completes in createdb.cmd
    A 'checkpoint' file is created (step1.chk, step2.chk...step9.chk)

```

As each checkpoint file is created  
    Display a checkmark on the panel next to the step just completed.  
When all 9 steps are complete.  
    Cleanup scripts and command files  
    exit

**' Panel 11b – Database Resizing in Process (frmResize)**

Display a checklist of steps to resize the database  
Execute the command file resizedb.cmd  
As each step completes in resizedb.cmd  
    A 'checkpoint' file is created (step1.chk)  
As each checkpoint file is created  
    Display a checkmark on the panel next to the step just completed.  
When all steps are complete.  
    Cleanup scripts and command files  
    exit

## Command Line Parameters

The following command line parameters are recognized by the brunner utility

### **/D**

Runs dbsizer in **development mode**. Development mode allows the user to modify the sizing formulas for tables and indexes as well as the Oracle engine parameters that are written to the INITPCPW.ORA file. In addition, the user is allowed to load and save multiple configuration files. (Note: When running in regular mode, only the configuration file default.mdb will be used.)

### **/DEBUG**

Runs dbsizer in **debug mode**. Normally as the Oracle utilities are executed, the command window which executes them is hidden from the user completely, including the task bar. If you run the wizard in debug mode, the command windows will only be minimized instead of hidden giving you the ability to see the command lines and any output from the utilities being executed.

## NT Server - Registry Entries

When the Oracle sizing wizard is run by the client to create their database, a number of entries are written to the NT Server's system registry. The following entries are created by dbSizer during database creation.

### KEYS USED BY THE HEALTH and WELL-BEING UTILITY (HWB)

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWora\LogFiles]
"Age"="90"
```

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWora\Extents]
"Number"="1"
```

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWora\HWB]
"Tables"="1"
"Performance"="0"
"Use Note of the Day"="True"
```

The **Age** key controls how long messages are kept in the brunner.log file. This value is set during install and there is no method for changing this value with the exception of using the regedit program supplied as part of the NT Server Operating System.

The **Number** key controls how many extents are required before HWB will attempt to automatically fix the table or index.

The last three are used by HWB to control whether or not **Tables** and/or **Performance** statistics are checked during execution. By default, tables are checked, performance is not. The **Note of the Day** entry determines whether or not HWB will report fatal errors back to the user via the T\_NOTE\_OF\_THE\_DAY table.

---

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWora\Keys]
"PCPAYSYS"="_=ë_Üh"
"INTERNAL"="Üä Y)_fö"
"MaintKey"="_=ë_Üh"
"MIGRATE"="re_ëY=Ä "
"SUPEROP"="_æö! _"
"REPORTS"=",_u%y_^*f"
"Default"="_=ë_Üh"
"SYS"="'_-C_L(Ö"
"SYSTEM"="f3TNäYIp"
```

These keys represent the user id's and passwords which can be part of a template (.brt file). In order to use one of the user id / password combinations, the user id must be surrounded by %'s in the .brt file. For example, to use the SrvMgr23 utility to run a SQL file named dothis.sql and use the INTERNAL id and password, the following line would be in the dothis.brt file.

```
connect INTERNAL / %INTERNAL%
...some sql code here
```

At run time, brunner will retrieve the value for the INTERNAL key from the registry, decode the key value and write the following to the tempn.sql file in the c:\temp folder

```
connect INTERNAL / THEPASSWORD
_some sql code here
```

---

```
{HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWora\Files}
"Home"="C:\\ORANT\\BIN"
"Maintenance"="C:\\ORADATA\\PCPW\\admin\\maint"
"Admin"="C:\\ORADATA\\PCPW\\ADMIN"
"Backup"=" " "
```

These settings let the Wizard, BRunner and HWB know where to find other files that they may need during execution

---

## brunner.exe

### Database Utility Program

#### Overview

The **brunner** utility is used by the client to perform the following database procedures

1. Manually bring the database up in normal or restricted mode
2. Manually shut the database down
3. Manually perform the database backup as established by the sizing wizard during database creation.
4. Manually reschedule the automated backup process as established by the sizing wizard during database creation.

The **brunner** utility is also used to perform some of these functions during the database creation process. In this mode, **brunner** is executed with command line parameters so that user intervention is not required. (See the **dbSizer.exe** detailed design spec, **dbSizer.doc**, for more information on the usage of **brunner** during database creation)

In general, regardless of which task **brunner** is performing the process is as follows;

1. check to see if the database is up or down.
2. if the function is passed on the command line, perform it...if not, display a menu of available functions based upon the current state of the database and let the user select which function to perform.
3. create a command file to perform the requested function (if SQL based, create the SQL file to perform the function and a command file to execute the SQL using the **SrvMgr23** utility supplied by Oracle)
4. delete the command file and the SQL file
5. exit

Some of the functions use pre-defined command file *templates* called .BRT files. These files are identical to the command files or SQL files that will be used to perform the various brunner functions, however they require that an Oracle password be supplied on the command line to the Oracle utility that is being executed. In order to hide the password, *placeholders* are used in the .BRT files and brunner will perform the following steps when executing a *secure batch file*.

1. open the batch template file (.brt)
2. create a temporary batch file (tempn.cmd) in the c:\temp folder
3. read each line from the template file
4. if the line contains a password placeholder, lookup the password in the system registry, decode it and place it in the temporary file, otherwise write the line as is to the temporary file.
5. execute the temporary file
6. delete the temporary file
7. exit

During execution, brunner maintains a log file which contains information about each run. Dates and times are written to the log along with the function which was requested and any errors that occurred during execution.

At any given time, the log file contains entries for the past 90 days. Log entries older than 90 days are *rolled off* the log. The number of days (90 is the default) worth of messages kept in the log file can be altered by changing an entry in the system registry. See the section on Registry entries for more information.

## Psedo-Code

Following is **pseudo-code** for the bunner utility program.

```
center the main form
if the command line contains "/MSG:"
    take the text that follows and display it on the screen in a message box
    exit
end if
get the location of the Oracle binaries from the registry
get the language setting from the registry
if the command line is NOT /SCHEDULE then
    check to see if the database is up or down (run checkdb.brt)
    if we can't determine the status of the database
        log the error
        exit
    end if
    display the appropriate bitmap on the form so the user knows the db status
end if
if no command line was specified
    display a menu of choices to the user
end if
write the start time and request to the log file
branch to the requested process

' backup branch
if the database is down, we can't perform the backup, so...
    log the error
    exit
end if
if we're using the copy or compress method
    make sure there's enough disk space on the destination drive
    if not
        log the error
        exit
    end if
    if the destination folder does not exist
        create it
    end if
end if
bookmark the Oracle alter log
perform the backup (run backup.brt which is created by dbsizer during install)
check the Oracle alter log for Oracle errors
if any errors
    write them to the brunner log
    write a Note of the Day entry to the database
end if
close the log file
exit
```

**' start the database (normal) branch**

bookmark the Oracle alert log  
start the database (run startdb.brt)  
check the Oracle alert log for errors  
if any errors  
    write them to the brunner log  
end if  
close the log file  
exit

**' stop the database branch**

**' parameter: RunStats**

if RunStats is true  
    update database statistics (run dopperf.sql)  
end if  
bookmark the Oracle alert log  
stop the database (run stopdb.brt)  
check the Oracle alert log for errors  
if any errors  
    write them to the brunner log  
end if  
close the log file  
exit

**' re-start the database branch**

*' difference between start and restart, is that restart clears any*

*' Note of the Day entry in the database. This is done in the*

*' restart.brt template file.*

bookmark the Oracle alert log  
start the database (run restartdb.brt)  
check the Oracle alert log for errors  
if any errors  
    write them to the brunner log  
end if  
close the log file  
exit

**' schedule branch**

check to see if there's already a call to BRUNNER with /SCHEDULE in the AT list  
if not  
    run schdback.cmd to schedule the backup process  
end if  
exit

**' start the database (restricted) branch**

bookmark the Oracle alert log  
start the database (run restrict.brt)  
check the Oracle alert log for errors  
if any errors  
    write them to the brunner log  
end if  
close the log file  
exit

## Command Line Parameters

The following command line parameters are recognized by the brunner utility

**/BACKUP**

causes brunner to execute the backup.brt file to perform the backup procedure

**/BACKUPSTOP**

same as /BACKUP, except it causes brunner to *update database statistics* (by running doperf.sql) before performing the backup.

**/MSG: msgText**

displays a dialog box with the text, *msgText*.

**/RESTRICT**

starts the database in *restricted* mode

**/SCHEDULE**

schedules the automated backup using NT's AT scheduler service. (runs the schdback.cmd command file.)

**/START**

starts the database in *normal* mode

**/STOP**

stops the database using the *immediate* mode

## NT Server - Registry Entries

When the Oracle sizing wizard is run by the client to create their database, a number of entries are written to the NT Server's system registry. The following entries are used by the brunner utility during execution

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWOr\LogFiles]
"Age"="90"
```

This key controls how long messages are kept in the brunner.log file. This value is set during install and there is no method for changing this value with the exception of using the regedit program supplied as part of the NT Server Operating System.

---

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWOr\Keys]
"PCPAYSYS"=" "
"INTERNAL"=" "
"MaintKey"=" "
"MIGRATE"="re ey=Ã "
"SUPEROP"=" "
"REPORTS"="u%y_ .f"
"Default"=" "
"SYS"=" "
"SYSTEM"="f3TNäYip"
```

These keys represent the user id's and passwords which can be part of a template (.brt) file. In order to use one of the user id / password combinations, the user id must be surrounded by %'s in the .brt file. For example, to use the SrvMgr23 utility to run a SQL file named dothis.sql and use the INTERNAL id and password, the following line would be in the dothis.brt file.

```
connect INTERNAL / %INTERNAL%
...some sql code here
```

At run time, brunner will retrieve the value for the INTERNAL key from the registry, decode the key value and write the following to the tempn.sql file in the c:\temp folder

```
connect INTERNAL / THEPASSWORD
...some sql code here
```

---

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWOr\Files]
"Home"="C:\\ORANT\\BIN"
"Maintenance"="C:\\ORADATA\\PCPW\\admin\\maint"
"Admin"="C:\\ORADATA\\PCPW\\ADMIN"
"Backup"=" "
```

These settings let brunner know where to find other files that it may need during execution

## Source Code

Following the source code for the brunner utility version 1.05-10.

```
VERSION 5.00
Begin VB.Form Form1
    Caption       = "ADP PC/Payroll Batch Runner"
    ClientHeight  = 3705
    ClientLeft    = 60
    ClientTop     = 345
    ClientWidth   = 5805
    Icon          = "Form1.frx":0000
    LinkTopic     = "Form1"
    ScaleHeight   = 3705
    ScaleWidth    = 5805
    StartUpPosition = 2 'CenterScreen
Begin VB.CommandButton Command1
    Caption       = "Close"
    Height        = 390
    Left          = 4515
    TabIndex      = 4
    Top           = 3135
    Visible       = 0 'False
    Width         = 1140
End
Begin VB.Timer Timer2
    Enabled       = 0 'False
    Interval      = 3000
    Left          = 4860
    Top           = 495
End
Begin VB.Timer Timer1
    Enabled       = 0 'False
    Interval      = 1000
    Left          = 4860
    Top           = 45
End
Begin VB.PictureBox Picture1
    BorderStyle   = 0 'None
    BeginProperty Font
        Name       = "Arial"
        Size        = 8.25
        Charset     = 0
        Weight      = 400
        Underline   = 0 'False
        Italic      = 0 'False
        Strikethrough = 0 'False
    EndProperty
    Height        = 3180
    Left          = 120
    Picture       = "Form1.frx":1CFA
    ScaleHeight   = 3180
    ScaleWidth    = 1830
    TabIndex      = 0
    Top           = 75
    Width         = 1830
End
Begin VB.Label Label3
    Alignment     = 2 'Center
    Caption       = "Process msg here"
    BeginProperty Font
        Name       = "Arial"
```

```

        Size           = 12
        Charset        = 0
        Weight         = 700
        Underline      = 0 'False
        Italic         = 0 'False
        Strikethrough  = 0 'False
    EndProperty
    Height            = 855
    Left              = 2055
    TabIndex          = 5
    Top               = 2160
    Visible           = 0 'False
    Width             = 3600
End
Begin VB.Label Label4
    Caption           = "Label4"
    Height            = 180
    Left              = 135
    TabIndex          = 3
    Top               = 3315
    Width             = 1395
End
Begin VB.Label Label2
    Caption           = "Process running:"
    BeginProperty Font
        Name           = "Arial"
        Size            = 9.75
        Charset         = 0
        Weight          = 700
        Underline       = 0 'False
        Italic          = 0 'False
        Strikethrough    = 0 'False
    EndProperty
    Height            = 300
    Left              = 2055
    TabIndex          = 2
    Top               = 120
    Width             = 3630
End
Begin VB.Label Label1
    Caption           = "Label1"
    Height            = 1230
    Left              = 2055
    TabIndex          = 1
    Top               = 435
    Width             = 3495
End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim startTime As Date
Dim runningProcess As String
Dim myTaskId As Long
Dim logFile As Integer
Dim g_user As Boolean

Dim g_MaintPassword As String
Dim g_BackupFolder As String

Dim lAlertLogLength As Double

Private Sub Command1_Click()
    End

```

End Sub

Private Sub Form\_Load()

Dim i As Integer

' -----  
' cleanup() function added to make sure any file remnants weren't  
' left behind from previous BRunner processes.  
' -----

' rc = Cleanup()

i = SetWindowPos(Me.hwnd, HWND\_TOPMOST, \_  
Me.Left \ Screen.TwipsPerPixelX, Me.Top \ Screen.TwipsPerPixelY, \_  
Me.Width \ Screen.TwipsPerPixelX, Me.Height \ Screen.TwipsPerPixelY, 0)

runningProcess = ""  
bProcess = False

If Mid\$(Command\$, 1, 5) = "/MSG:" Then  
cmdLine = Command\$  
frmMessage.Show vbModal  
End If

g\_szOracleHome = GetSetting("PCPWora", "Files", "Home", "")

' -----  
' get the language  
' -----

g\_LANGUAGE = RegGetValue(HKEY\_CURRENT\_USER, "Control Panel\International", "Locale")

Select Case g\_LANGUAGE  
Case "00001009"  
g\_LANGOFFSET = 1000  
Case "00000C0C"  
g\_LANGOFFSET = 2000  
Case Else  
g\_LANGOFFSET = 0  
End Select

On Error GoTo NoLanguageRes

txt\$ = RES(101)  
GoTo LanguageContinue

NoLanguageRes:

g\_LANGOFFSET = 0

LanguageContinue:

On Error GoTo 0

' DEBUG: uncomment the next line to force language selection  
' g\_LANGOFFSET = 2000

Label4.Caption = "v" + Format\$(App.Major) + "." + Format\$(App.Minor, "00")  
Label1.Caption = ""  
Label2.Caption = RES(103)  
Command1.Caption = RES(104)

Me.Caption = RES(102)

g\_MaintPassword = GetSetting("PCPWora", "Keys", "MaintKey", "")  
g\_MaintPassword = StrDecode(g\_MaintPassword, 14755)

g\_BackupFolder = GetSetting("PCPWora", "Files", "Backup", "")

cmdLine = Command\$

```

If cmdLine <> "/SCHEDULE" Then

    ' check to see if the database is up or down
    rc = ExecuteSecureBatchFile(App.Path & "\checkdb.brt", True)
    fh = FreeFile
    g_dbopen = True
    On Error GoTo NoOutFile
    Open App.Path & "\checkdb.out" For Input As #fh
    Do Until EOF(fh)
        Line Input #fh, tbuf$
        If UCase(Left(tbuf$, 4)) = "ORA-" Then
            g_dbopen = False
        End If
    Loop
    Close #fh
    On Error GoTo 0

    If g_dbopen Then
        Picture1.Picture = LoadPicture(App.Path & "\images\dbup.bmp")
    Else
        Picture1.Picture = LoadPicture(App.Path & "\images\dbdown.bmp")
    End If

End If

' -----
' test code to set the command line parameter
' comment the following line before building
' -----
' cmdLine = "/MSG:This is a test"

g_user = False
If cmdLine = "" Then
    g_user = True
    frmGetCommand.Show vbModal
End If

g_MaintPassword = GetSetting("PCPWOr", "Keys", "MaintKey", "")
g_MaintPassword = StrDecode(g_MaintPassword, 14755)

g_szMaint = GetSetting("PCPWOr", "Files", "Maintenance", "")

startTime = Now

rc = OpenLogFile(App.Path & "\brunner", 0)
rc = WriteLogFile(RES(203))
rc = WriteLogFile(RES(204) & cmdLine)

Select Case UCase$(cmdLine)
    Case "/BACKUP"
        Label1.Caption = RES(108)
    Case "/START"
        Label1.Caption = RES(109)
    Case "/STOP"
        Label1.Caption = RES(110)
    Case "/SCHEDULE"
        Label1.Caption = RES(111)
End Select

Refresh
Timer2.Enabled = True

Exit Sub

NoOutFile:
Refresh

```

```

MsgBox RES(125), vbOKOnly + vbCritical, RES(102)
End

End Sub

Private Function DoBackup()

    Dim fhErr As Integer
    Dim fh As Integer
    Dim bytesneeded As Double
    Dim bytesavailable As Double

    ' if the database is down, don't let the backup take place
    If g_dbopen = False Then
        rc = WriteLogFile(RES(205))
        rc = CloseLogFile()
    End
End If

    ' if copy or compress method, make sure there's enough disk space
    ' before attempting a backup
    fh = FreeFile
    Open App.Path & "\size.bat" For Output As #fh
    Print #fh, "set ORACLE_SID=PCPW"
    Print #fh, g_szOracleHome & "\sqlplus pcpsys/" & g_MaintPassword & " @" & App.Path &
"\size.sql " & App.Path
    Close #fh

    rc = RemoveFile(App.Path & "\sizedone.out")
    rc = ExecuteSecureBatchFile(App.Path & "\size.bat", False)

    ' make sure the prior step is complete before continuing
    Do Until Dir$(App.Path & "\sizedone.out", vbNormal) <> ""
        DoEvents
    Loop
    rc = RemoveFile(App.Path & "\size.bat")

    fh = FreeFile
    Open App.Path & "\size.out" For Input As #fh
    Line Input #fh, tempbuf$
    bytesneeded = Val(tempbuf$)
    Close #fh

    ' make sure the backup folder exists
    g_BackupFolder = Trim(g_BackupFolder)
    If g_BackupFolder <> "" Then
        If Dir$(g_BackupFolder, vbDirectory) = "" Then
            MkDir g_BackupFolder
        End If
        bytesavailable = GetDiskFreeSpaceLarge(Mid$(g_BackupFolder, 1, 1) & ":\")
        If bytesneeded > bytesavailable Then
            rc = WriteLogFile(RES(206) & Mid$(g_BackupFolder, 1, 1) & ":\ " & RES(207))
            ' debug code starts here
            rc = WriteLogFile("Bytes needed: " & bytesneeded & " Bytes Available on " &
Mid$(g_BackupFolder, 1, 1) & ":\ " & bytesavailable)
            ' debug code ends here
            rc = WriteNoteOfTheDay(RES(208) & Mid$(g_BackupFolder, 1, 1) & ":\ " & RES(130))
            rc = CloseLogFile()
        End
    End If
End If

    rc = WriteLogFile(RES(209))

    ' -----
    ' before performing the backup, run the perf.sql script that resides in the

```

```

' \oradata\pcpw\admin folder. This will generate a perf.out report. Open the report
' and add it to the perfsumm.out file.
' -----
rc = WriteLogFile(RES(210))
GetPerfStats
rc = WriteLogFile(RES(211))

rc = ExecuteSecureBatchFile(App.Path & "\backup.brc", False)

rc = WriteLogFile(RES(212))

If g_user Then
    MsgBox RES(112), vbOKOnly + vbInformation, RES(102)
End If

' -----
' TODO: check Oracle Alert log for messages
' -----
startPoint = Val(GetSetting("PCPWOr", "ALERT Log", "LastOffset", "1"))

If Dir$(g_szMaint & "..\oraerr.lst", vbNormal) = "" Then
    GoTo MissingControlFile
End If

fhErr = FreeFile
Open g_szMaint & "..\oraerr.lst" For Input As #fhErr

fhIn = FreeFile
rc = WriteLogFile(RES(213) & startPoint & RES(214))
Open g_szMaint & "..\..\log\pcpwALRT.log" For Input As #fhIn
Seek #fhIn, startPoint
Do Until EOF(fhIn)
    Line Input #fhIn, tbuf$

    Seek #fhErr, 1
    Line Input #fhErr, oraErr$
    Do Until EOF(fhErr)
        If InStr(UCase$(tbuf$), UCase$(Trim(oraErr$))) Then
            rc = WriteLogFile(RES(137) & oraErr$ & RES(138))
            rc = WriteNoteOfTheDay(RES(139) & oraErr$ & RES(140))
        End If
        Line Input #fhErr, oraErr$
    Loop

Loop

Close #fhIn

Close #fhErr

lAlertLogLength = FileLen(g_szMaint & "..\..\log\pcpwALRT.log")
SaveSetting "PCPWOr", "ALERT Log", "LastOffset", Format$(lAlertLogLength)
GoTo AlertLogChecked

MissingControlFile:
    rc = WriteLogFile(RES(141))
    rc = WriteNoteOfTheDay(RES(142))

AlertLogChecked:

    rc = CloseLogFile()
End

End Function

Private Function DoStartDb()

    rc = WriteLogFile(RES(143))

```

```

lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")

rc = ExecuteSecureBatchFile(App.Path & "\startdb.brt", False)
' -----
' Make sure the database is up in normal mode
' -----
fhCheck = FreeFile
Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
Seek #fhCheck, lAlertLogLength
bClosed = False
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$
    If UCCase$(Trim(buf$)) = UCCase$("Completed: alter database open") Then
        bClosed = True
    End If
    If UCCase$(Trim(buf$)) = UCCase$("Completed: alter database pay4win open") Then
        bClosed = True
    End If
    If UCCase$(Trim(buf$)) = UCCase$("Completed: alter database " & Chr$(34) &
"pay4win" & Chr$(34) & " open") Then
        bClosed = True
    End If
    DoEvents
Loop
Close #fhCheck

If bClosed = False Then
    ' -----
    ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
    ' table, then get out.
    ' -----
    rc = WriteLogFile(RES(144))
    If g_user Then
        Label3.FontSize = 9
        Label3.Caption = RES(116) & Chr$(10) & RES(115)
        Label3.Visible = True
        Command1.Visible = True
        Me.Refresh
    End If
Else
    If g_user Then
        Picture1.Picture = LoadPicture(App.Path & "\images\dbup.bmp")
        Label3.FontSize = 12
        Label3.Caption = RES(113)
        Label3.Visible = True
        Command1.Visible = True
        Me.Refresh
    End If
End If

rc = WriteLogFile(RES(145))

rc = CloseLogFile()
If Not g_user Then
    End
End If

End Function

Private Function DoStopDb(RunStats As Boolean)

    If RunStats Then
        rc = WriteLogFile(RES(146))
        ' -----
        ' before performing the backup, run the perf.sql script that resides in the
        ' \oradata\pcpw\admin folder. This will generate a perf.out report. Open the

```

```

report
    ' and add it to the perfsumm.out file.
    ' -----
    GetPerfStats
    rc = WriteLogFile(RES(147))
End If

    ' -----
    ' Before shutting down, get the length of the alert log
    ' so I don't have to read the whole thing to get to the
    ' end
    ' -----
    lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")

    rc = WriteLogFile(RES(148))

    rc = ExecuteSecureBatchFile(App.Path + "\stopdb.brt", False)

    ' -----
    ' now that the database is shutdown, make sure the shutdown
    ' was successful and without errors
    ' -----
    fhCheck = FreeFile
    Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
    Seek #fhCheck, lAlertLogLength
    bClosed = False
    Do Until EOF(fhCheck)
        Line Input #fhCheck, buf$
        If UCase$(Trim(Mid$(buf$, 1, 25))) = UCase$("Completed: ALTER DATABASE") Then
            bClosed = True
        End If
        DoEvents
    Loop
    Close #fhCheck

    If bClosed = False Then
        ' -----
        ' this means the database was not shutdown properly
        ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
        ' table, then get out.
        ' -----
        rc = WriteLogFile(RES(149))
        If g_user Then
            Label3.FontSize = 9
            Label3.Caption = RES(114) & Chr$(10) & RES(115)
            Label3.Visible = True
            Command1.Visible = True
            Me.Refresh
        End If
    Else
        rc = WriteLogFile(RES(150))
        If g_user Then
            Picture1.Picture = LoadPicture(App.Path & "\images\dbdown.bmp")
            Label3.FontSize = 12
            Label3.Caption = RES(117)
            Label3.Visible = True
            Command1.Visible = True
            Me.Refresh
        End If
    End If

    rc = CloseLogFile()
    If Not g_user Then
        End
    End If

End Function

```

```

Private Function DoRestart()

    lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")

    rc = WriteLogFile(RES(151))

    rc = ExecuteSecureBatchFile(App.Path + "\restart.brt", False)

    rc = WriteLogFile(RES(152))

    ' -----
    ' Make sure the database is up in normal mode
    ' -----
    fhCheck = FreeFile
    Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
    Seek #fhCheck, lAlertLogLength
    bClosed = False
    Do Until EOF(fhCheck)
        Line Input #fhCheck, buf$
        If UCase$(Trim(buf$)) = UCase$("Completed: alter database open") Then
            bClosed = True
        End If
        If UCase$(Trim(buf$)) = UCase$("Completed: alter database pay4win open") Then
            bClosed = True
        End If
        If UCase$(Trim(buf$)) = UCase$("Completed: alter database " & Chr$(34) &
"pay4win" & Chr$(34) & " open") Then
            bClosed = True
        End If
        DoEvents
    Loop
    Close #fhCheck

    If bClosed = False Then
        ' -----
        ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
        ' table, then get out.
        ' -----
        rc = WriteLogFile(RES(153))
        If g_user Then
            Label3.FontSize = 9
            Label3.Caption = RES(116) & Chr$(10) & RES(115)
            Label3.Visible = True
            Command1.Visible = True
            Me.Refresh
        End If
    Else
        If g_user Then
            Picture1.Picture = LoadPicture(App.Path & "\images\dbup.bmp")
            Label3.FontSize = 12
            Label3.Caption = RES(113)
            Label3.Visible = True
            Command1.Visible = True
            Me.Refresh
        End If
    End If

    rc = WriteLogFile(RES(154))

    rc = CloseLogFile()
    If Not g_user Then
        End
    End If

End Function

```

```

Private Function DoSchedule()

    rc = WriteLogFile(RES(155))
    If Dir$(App.Path & "\schdback.cmd", vbNormal) = "" Then
        rc = WriteLogFile(RES(156))
        rc = WriteLogFile(RES(157))
        If g_user Then
            MsgBox RES(118), vbOKOnly + vbInformation, RES(102)
        End If
        rc = CloseLogFile()
        End
    Else
        ' check to see if the scheduler already contains an entry for
        ' BRUNNER to backup the database
        bProcess = True
        ExecDOSCmd ("cmd /c net start schedule")
        bProcess = False
        bProcess = True
        ExecDOSCmd ("cmd /c at > c:\temp\at.txt")
        bProcess = False

        fhIn = FreeFile
        On Error GoTo NoAtFile
        Open "c:\temp\at.txt" For Input As #fhIn
        found = False
        Do While Not EOF(fhIn)
            Line Input #fhIn, buf
            offset = InStr(buf, "BRUNNER.EXE /BACKUP")
            If offset > 0 Then
                found = True
            End If
        Loop
        Close #fhIn
        GoTo AtFileOkay

    NoAtFile:
        rc = WriteLogFile(RES(158))
        On Error GoTo 0
        GoTo ExitPoint

    AtFileOkay:

        If found = False Then
            ' now execute the temporary batch file
            bProcess = True
            rc = ExecuteSecureBatchFile(App.Path & "\schdback.cmd", False)
            bProcess = False
            rc = WriteLogFile(RES(159))
        Else
            rc = WriteLogFile(RES(160))
        End If
        End If

    ExitPoint:

        If Dir$("c:\temp\at.txt", vbNormal) <> "" Then
            Kill "c:\temp\at.txt"
        End If

        On Error GoTo 0

        If dirCreated Then
            Rmdir "c:\temp"
        End If

        If g_user Then

```

```

        MsgBox RES(119), vbOKOnly + vbInformation, RES(102)
    End If

    rc = CloseLogFile()
End

End Function

Private Function DoRestrict()

    rc = WriteLogFile(RES(161))

    rc = ExecuteSecureBatchFile(App.Path + "\restrict.brt", False)

    rc = WriteLogFile(RES(162))

    ' -----
    ' Make sure the database is up in normal mode
    ' -----
    fhCheck = FreeFile
    Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
    Seek #fhCheck, 1AlertLogLength
    bClosed = False
    Do Until EOF(fhCheck)
        Line Input #fhCheck, buf$
        If UCase$(Trim(buf$)) = UCase$("Completed: alter database open") Then
            bClosed = True
        End If
        If UCase$(Trim(buf$)) = UCase$("Completed: alter database pay4win open") Then
            bClosed = True
        End If
        If UCase$(Trim(buf$)) = UCase$("Completed: alter database " & Chr$(34) &
"pay4win" & Chr$(34) & " open") Then
            bClosed = True
        End If
        DoEvents
    Loop
    Close #fhCheck

    If bClosed = False Then
        ' -----
        ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
        ' table, then get out.
        ' -----
        rc = WriteLogFile(RES(153))
        If g_user Then
            Label3.FontSize = 9
            Label3.Caption = RES(116) & Chr$(10) & RES(115)
            Label3.Visible = True
            Command1.Visible = True
            Me.Refresh
        End If
    Else
        If g_user Then
            Picture1.Picture = LoadPicture(App.Path & "\images\dbup.bmp")
            Label3.FontSize = 12
            Label3.Caption = RES(113)
            Label3.Visible = True
            Command1.Visible = True
            Me.Refresh
        End If
    End If

    rc = WriteLogFile(RES(154))

    rc = CloseLogFile()
    If Not g_user Then

```

```

        End
    End If

End Function

Private Sub Timer2_Timer()

    Timer2.Enabled = False
    Select Case UCase$(cmdLine)
        Case "/BACKUP"
            Call DoBackup
        Case "/RESTART"
            Call DoRestart
        Case "/START"
            Call DoStartDb
        Case "/STOP"
            Call DoStopDb(False)
        Case "/BACKUPSTOP"
            Call DoStopDb(True)
        Case "/SCHEDULE"
            Call DoSchedule
        Case "/RESTRICT"
            Call DoRestrict
    End Select

End Sub

Public Function RemoveFile(szFile As String) As Boolean

    On Error GoTo CannotRemoveFile
    If Dir$(szFile, vbNormal) <> "" Then
        Kill szFile
    End If
    On Error GoTo 0
    RemoveFile = True
    Exit Function

CannotRemoveFile:
    On Error GoTo 0
    RemoveFile = False
    Exit Function

End Function

Public Function GetPerfStats() As Boolean

    Dim fh As Integer
    Dim t1 As String
    Dim t2 As String

    t1 = GetSetting("PCPW0ra", "Keys", "MaintKey", "")
    t2 = StrDecode(t1, 14755)

    fh = FreeFile
    Open App.Path & "\maint\doperf.sql" For Output As #fh
    Print #fh, "connect pcpayssys/" & t2 & ";"
    Print #fh, "execute updperfstat;"
    Print #fh, "exit;"
    Close #fh

    fh = FreeFile
    Open App.Path & "\maint\doperf.bat" For Output As #fh
    Print #fh, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\doperf.sql"
    Close #fh

    ExecDOSCmd (App.Path & "\maint\doperf.bat")

```

```

    rc = RemoveFile(App.Path & "\maint\doperf.bat')
    rc = RemoveFile(App.Path & "\maint\doperf.sql")
    GetPerfStats = True

End Function

Private Function Cleanup() As Boolean

    On Error Resume Next
    Kill "c:\temp\tmp*.cmd"
    On Error GoTo 0

End Function

VERSION 5.00
Begin VB.Form frmGetCommand
    Caption           = "ADP PC/Payroll Batch Runner"
    ClientHeight      = 3885
    ClientLeft        = 60
    ClientTop         = 345
    ClientWidth       = 5805
    LinkTopic         = "Form2"
    ScaleHeight       = 3885
    ScaleWidth        = 5805
    StartUpPosition  = 3 'Windows Default
    Begin VB.CommandButton Command2
        Caption       = "Close"
        Height        = 405
        Left          = 4590
        TabIndex      = 4
        Top           = 3270
        Width         = 990
    End
    Begin VB.CommandButton Command1
        Caption       = "&Run"
        Height        = 405
        Left          = 3510
        TabIndex      = 3
        Top           = 3255
        Width         = 990
    End
    Begin VB.ComboBox Combo1
        Height        = 315
        ItemData      = "frmGetCommand.frx":0000
        Left          = 2040
        List          = "frmGetCommand.frx":0002
        Style         = 2 'Dropdown List
        TabIndex      = 2
        Top           = 435
        Width         = 3615
    End
    Begin VB.PictureBox Picture1
        BorderStyle   = 0 'None
        BeginProperty Font
            Name       = "Arial"
            Size      = 8.25
            Charset    = 0
            Weight     = 400
            Underline  = 0 'False
            Italic     = 0 'False
            Strikethrough = 0 'False
        EndProperty
        Height        = 3180
        Left          = 105
        Picture       = "frmGetCommand.frx":0004
    End

```

```

        ScaleHeight      = 3180
        ScaleWidth       = 1830
        TabIndex         = 0
        Top              = 120
        Width            = 1830
    End
    Begin VB.Label Label4
        Caption           = "Label4"
        Height            = 180
        Left              = 135
        TabIndex         = 5
        Top              = 3390
        Width            = 1395
    End
    Begin VB.Label Label2
        Caption           = "Select process"
        BeginProperty Font
            Name           = "Arial"
            Size           = 9.75
            Charset        = 0
            Weight         = 700
            Underline      = 0 'False
            Italic         = 0 'False
            Strikethrough   = 0 'False
        EndProperty
        Height            = 300
        Left              = 2025
        TabIndex         = 1
        Top              = 135
        Width            = 3630
    End
End
Attribute VB_Name = "frmGetCommand"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Command1_Click()

    Select Case Combol.List(Combol.ListIndex)
        Case RES(121)
            cmdLine = "/BACKUP"
        Case RES(120)
            cmdLine = "/STOP"
        Case RES(123)
            cmdLine = "/START"
        Case RES(122)
            cmdLine = "/SCHEDULE"
        Case RES(124)
            cmdLine = "/RESTRICT"
    End Select
    Unload Me

End Sub

Private Sub Command2_Click()

    End

End Sub

Private Sub Form_Load()

    Dim i As Integer

    i = SetWindowPos(Me.hWnd, HWND_TOPMOST, _
        Me.Left \ Screen.TwipsPerPixelX, Me.Top \ Screen.TwipsPerPixelY, _

```

```

Me.Width \ Screen.TwipsPerPixelX, Me.Height \ Screen.TwipsPerPixelY, 0)

If g_dbopen Then
    Picture1.Picture = LoadPicture(App.Path & "\images\dbup.bmp")
Else
    Picture1.Picture = LoadPicture(App.Path & "\images\dbdown.bmp")
End If

Me.Left = (Screen.Width - Me.ScaleWidth) / 2
Me.Top = (Screen.Height - Me.ScaleHeight) / 2
Me.Caption = RES(102)
Label2.Caption = RES(105)
Command1.Caption = RES(106)
Command2.Caption = RES(104)

Label4.Caption = "v" + Format$(App.Major) + "." + Format$(App.Minor, "00")

If g_dbopen Then
    Combo1.AddItem RES(120)
    Combo1.AddItem RES(121)
    Combo1.AddItem RES(122)
Else
    Combo1.AddItem RES(123)
    Combo1.AddItem RES(124)
    Combo1.AddItem RES(122)
End If

Combo1.ListIndex = 0

End Sub

```

```

VERSION 5.00
Begin VB.Form frmMessage
    Caption           = "ADP PC/Payroll Batch Runner"
    ClientHeight      = 3705
    ClientLeft        = 60
    ClientTop         = 345
    ClientWidth       = 5805
    LinkTopic         = "Form1"
    ScaleHeight       = 3705
    ScaleWidth        = 5805
    StartUpPosition   = 3 'Windows Default
Begin VB.CommandButton Command2
    Caption           = "Close"
    Height            = 405
    Left              = 4620
    TabIndex          = 3
    Top               = 3060
    Width             = 990
End
Begin VB.PictureBox Picture1
    BorderStyle       = 0 'None
    BeginProperty Font
        Name           = "Arial"
        Size            = 8.25
        Charset         = 0
        Weight          = 400
        Underline       = 0 'False
        Italic          = 0 'False
        Strikethrough    = 0 'False
    EndProperty
    Height            = 3180
    Left              = 120
    Picture            = "frmMessage.frx":0000
    ScaleHeight       = 3180
    ScaleWidth        = 1830
    TabIndex          = 0
    Top               = 75
    Width             = 1830
End
Begin VB.Label Label4
    Caption           = "Label4"
    Height            = 180
    Left              = 120
    TabIndex          = 4
    Top               = 3345
    Width             = 1395
End
Begin VB.Label Label2
    Caption           = "Message:"
    BeginProperty Font
        Name           = "Arial"
        Size            = 9.75
        Charset         = 0
        Weight          = 700
        Underline       = 0 'False
        Italic          = 0 'False
        Strikethrough    = 0 'False
    EndProperty
    Height            = 300
    Left              = 2115
    TabIndex          = 2
    Top               = 75
    Width             = 3630
End
Begin VB.Label Label1
    Caption           = "Label1"
    Height            = 1800

```

```

        Left           = 2115
        TabIndex       = 1
        Top            = 360
        Width          = 3510
    End
End
Attribute VB_Name = "frmMessage"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim startTime As Date
Dim runningProcess As String
Dim myTaskId As Long

Private Sub Command2_Click()
    End
End Sub

Private Sub Form_Load()

    Dim i As Integer

    Me.Left = (Screen.Width - Me.ScaleWidth) / 2
    Me.Top = (Screen.Height - Me.ScaleHeight) / 2

    i = SetWindowPos(Me.hWnd, HWND_TOPMOST, _
        Me.Left \ Screen.TwipsPerPixelX, Me.Top \ Screen.TwipsPerPixelY, _
        Me.Width \ Screen.TwipsPerPixelX, Me.Height \ Screen.TwipsPerPixelY, 0)

    Me.Caption = RES(102)
    Label2.Caption = RES(107)
    Command2.Caption = RES(104)

    'If g_dbopen Then
    '    Picture1.Picture = LoadPicture(App.Path & "\images\dbup.bmp")
    'Else
    '    Picture1.Picture = LoadPicture(App.Path & "\images\dbdown.bmp")
    'End If

    Label4.Caption = "v" + Format$(App.Major) + "." + Format$(App.Minor, "00")

    Label1.Caption = Mid$(cmdLine, 6)

End Sub

Attribute VB_Name = "Module2"
Private Type STARTUPINFO
    cb As Long
    lpReserved As String
    lpDesktop As String
    lpTitle As String
    dwX As Long
    dwY As Long
    dwXSize As Long
    dwYSize As Long
    dwXCountChars As Long
    dwYCountChars As Long
    dwFillAttribute As Long
    dwFlags As Long
    wShowWindow As Integer
    cbReserved2 As Integer
    lpReserved2 As Long
    hStdInput As Long
    hStdOutput As Long
    hStdError As Long

```

```

End Type

Private Type PROCESS_INFORMATION
    hProcess As Long
    hThread As Long
    dwProcessID As Long
    dwThreadID As Long
End Type

Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal _
    hHandle As Long, ByVal dwMilliseconds As Long) As Long

Private Declare Function CreateProcessA Lib "kernel32" (ByVal _
    lpApplicationName As Long, ByVal lpCommandLine As String, ByVal _
    lpProcessAttributes As Long, ByVal lpThreadAttributes As Long, _
    ByVal bInheritHandles As Long, ByVal dwCreationFlags As Long, _
    ByVal lpEnvironment As Long, ByVal lpCurrentDirectory As Long, _
    lpStartupInfo As STARTUPINFO, lpProcessInformation As _
    PROCESS_INFORMATION) As Long

Private Declare Function CloseHandle Lib "kernel32" (ByVal _
    hObject As Long) As Long

Private Const NORMAL_PRIORITY_CLASS = &H20&
Private Const INFINITE = -1&
Public Const SW_HIDE = 0
Public Const SW_MINIMIZE = 6
Public Const STARTF_USESHOWWINDOW = &H1

Public Sub ExecDOSCmd(cmdLine$)
    Dim proc As PROCESS_INFORMATION
    Dim start As STARTUPINFO

    ' Initialize the STARTUPINFO structure:
    start.cb = Len(start)
    start.wShowWindow = SW_HIDE
    start.dwFlags = STARTF_USESHOWWINDOW

    ' Start the shelled application:
    ret% = CreateProcessA(0&, cmdLine$, 0&, 0&, 1&, _
        NORMAL_PRIORITY_CLASS, 0&, 0&, start, proc)

    ' Wait for the shelled application to finish:
    ret% = WaitForSingleObject(proc.hProcess, INFINITE)
    ret% = CloseHandle(proc.hProcess)
End Sub

Attribute VB_Name = "Module1"
Public Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, _
    ByVal hWndInsertAfter As Integer, _
    ByVal X As Integer, ByVal Y As Integer, _
    ByVal cx As Integer, ByVal cy As Integer, _
    ByVal wFlags As Integer) As Integer

Global Const SWP_NOMOVE = 2
Global Const SWP_NOSIZE = 1
Global Const WndFlags = SWP_NOMOVE Or SWP_NOSIZE
Global Const HWND_TOPMOST = -1
Global Const HWND_NOTOPMOST = -2

Global bProcess As Boolean
Global cmdLine As String
Global g_szOracleHome As String
Global g_szMaint As String
Global g_MaintPassword As String

```

```

Global g_dbopen As Boolean
Global g_LANGUAGE As String
Global g_LANGOFFSET As Integer

Public Function CancelProcess()
    End
End Function

Public Function WriteNoteOfTheDay(szMsg As String) As Boolean

    Dim fh As Integer

    fh = FreeFile
    Open App.Path & "\maint\notd.sql" For Output As #fh
    Print #fh, "connect pcpaysys/%%PCPAYSYS%%;"
    Print #fh, "execute p_modify_postnote('" & szMsg & "','ADD');"
    Print #fh, "exit;"
    Close #fh

    rc = ExecuteSecureBatchFile(App.Path & "\maint\notd.sql", True)

    rc = RemoveFile(App.Path & "\maint\notd.sql")

    WriteNoteOfTheDay = True

End Function

Public Function ExecuteSecureBatchFile(szFile As String, bAsSQL As Boolean) As Boolean

' -----
' This function executes a batch file that contains passwords
' the batch file is opened, and copied to a temp location
' with %password% substitution so that the passwords are not
' exposed in the batch files that are persistent on the server
' -----

Dim fh As Integer
Dim fh2 As Integer
Dim fh3 As Integer
Dim tmpname As String

i = 0
tmpname = "c:\temp\tmp" & i & "%%%"
Do Until Dir$(tmpname & ".cmd", vbNormal) = ""
    i = i + 1
    tmpname = "c:\temp\tmp" & i & "%%%"
Loop

' create temporary batch file
If Dir$("c:\temp", vbDirectory) = "" Then
    MkDir ("c:\temp")
    dirCreated = True
Else
    dirCreated = False
End If

fh = FreeFile
Open szFile For Input As #fh
fh2 = FreeFile
If bAsSQL Then
    Open tmpname & ".sql" For Output As #fh2
    fh3 = FreeFile
    Open tmpname & ".cmd" For Output As #fh3
Else
    Open tmpname & ".cmd" For Output As #fh2
End If

Do While Not EOF(fh)

```

```

Line Input #fh, buf

' look for password placeholders
offset = InStr(buf, "%%PCPAYSYS%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "PCPAYSYS", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 12)
End If

offset = InStr(buf, "%%MIGRATE%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "MIGRATE", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 11)
End If

offset = InStr(buf, "%%MAINTKEY%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "MAINTKEY", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 12)
End If

offset = InStr(buf, "%%SUPEROP%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "SUPEROP", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 11)
End If

offset = InStr(buf, "%%REPORTS%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "REPORTS", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 11)
End If

offset = InStr(buf, "%%PASSWORD%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "Default", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 12)
End If

offset = InStr(buf, "%%INTERNAL%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "INTERNAL", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 12)
End If

offset = InStr(buf, "%%SYS%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "SYS", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 12)
End If

offset = InStr(buf, "%%SYSTEM%%")
If offset > 0 Then
    pw$ = GetSetting("PCPWOr", "Keys", "SYSTEM", "")
    pw$ = StrDecode(pw$, 14755)
    buf = Mid$(buf, 1, offset - 1) & pw$ & Mid$(buf, offset + 12)
End If

```

```

        Print #fh2, buf

Loop

Close #fh2
Close #fh

' now execute the temporary batch file
If bAsSQL Then
    Print #fh3, g_szOracleHome & "\SVRMGR23 @" & tmpname & ".SQL"
    Close #fh3
    ExecDOSCmd (tmpname & ".cmd")
    Kill tmpname & ".sql"
Else
    ExecDOSCmd (tmpname & ".cmd")
End If

Kill tmpname & ".cmd"
If dirCreated Then
    Rmdir "c:\temp"
End If

ExecuteSecureBatchFile = True

End Function

Public Function RES(resID As Integer) As String
    RES = LoadResString(g_LANGOFFSET + resID)
End Function

Public Function GetDiskFreeSpaceLarge(DriveLetter As String) As Double

    Dim hdb As Integer
    Dim bf As Double
    Dim buf As String
    Dim buf2 As String
    Dim fh As Integer

    fh = FreeFile
    On Error GoTo CantWrite
    Open DriveLetter + "test.txt" For Output As #fh
    On Error GoTo 0
    Print #fh, "Testing"
    Close #fh

    ' -----
    ' execute both command.com and cmd.com. If running on Win95
    ' the command.com will work, and cmd.com will fail. On WinNT4.0
    ' both will work, but the correct output of cmd.com will overwrite
    ' the incorrect output of command.com. This way the end result
    ' will be correct regardless of OS...
    ' -----
    ExecDOSCmd ("command.com /c dir " + DriveLetter + "test.txt > c:\dbsizer.lst")
    ExecDOSCmd ("cmd /c dir " + DriveLetter + "test.txt > c:\dbsizer.lst")
    hdb = FreeFile
    Open "c:\dbsizer.lst" For Input As #hdb
    Do Until EOF(hdb)
        Line Input #hdb, buf
        idx = InStr(UCase$(buf), UCase$(RES(215)))
        If idx > 0 Then
            buf = Left$(buf, idx - 2)
            For i% = Len(buf) To 1 Step -1
                If Mid$(buf, i%, 1) = " " Then
                    buf = Mid$(buf, i% + 1)
                    Exit For
                End If
            Next i%
        End If
    Loop

```

```

        End If
    Loop
    Close #hdb
    rc = RemoveFile("c:\dbsizer.lst")
    rc = RemoveFile(DriveLetter & "test.txt")

    buf2 = ""
    For i% = 1 To Len(buf)
        thischar = Mid$(buf, i%, 1)
        If thischar <> " " And thischar <> RES(216) Then
            buf2 = buf2 + thischar
        End If
    Next i%

    GetDiskFreeSpaceLarge = Val(buf2)
    Exit Function

CantWrite:
    On Error GoTo 0
    GetDiskFreeSpaceLarge = 0

End Function

Public Function RemoveFile(szFile As String) As Boolean

    On Error GoTo CannotRemoveFile
    If Dir$(szFile, vbNormal) <> "" Then
        Kill szFile
    End If
    On Error GoTo 0
    RemoveFile = True
    Exit Function

CannotRemoveFile:
    On Error GoTo 0
    RemoveFile = False
    Exit Function

End Function

Function StrEncode(s As String, key As Long) As String

'Written by Gary Ardell.
'free from all copyright restrictions

Dim N As Long, i As Long, ss As String
Dim k1 As Long, k2 As Long, k3 As Long, k4 As Long, t As Long
Dim salt As Boolean
Static saltvalue As String * 4

salt = False

If salt Then
    For i = 1 To 4
        t = 100 * (1 + Asc(Mid(saltvalue, i, 1))) * Rnd() * (Timer + 1)
        Mid(saltvalue, i, 1) = Chr(t Mod 256)
    Next
    s = Mid(saltvalue, 1, 2) & s & Mid(saltvalue, 3, 2)
End If

N = Len(s)
ss = Space(N)
ReDim sn(N) As Long

k1 = 11 + (key Mod 233): k2 = 7 + (key Mod 239)
k3 = 5 + (key Mod 241): k4 = 3 + (key Mod 251)

```

```

For i = 1 To N: sn(i) = Asc(Mid(s, i, 1)): Next i

For i = 2 To N: sn(i) = sn(i) Xor sn(i - 1) Xor ((k1 * sn(i - 1)) Mod 256): Next
For i = N - 1 To 1 Step -1: sn(i) = sn(i) Xor sn(i + 1) Xor (k2 * sn(i + 1)) Mod 256: Next
For i = 3 To N: sn(i) = sn(i) Xor sn(i - 2) Xor (k3 * sn(i - 1)) Mod 256: Next
For i = N - 2 To 1 Step -1: sn(i) = sn(i) Xor sn(i + 2) Xor (k4 * sn(i + 1)) Mod 256: Next

For i = 1 To N: Mid(ss, i, 1) = Chr(sn(i)): Next i

StrEncode = ss
saltvalue = Mid(ss, Len(ss) / 2, 4)

End Function

Function StrDecode(s As String, key As Long) As String

'Written by Gary Ardell.
'free from all copyright restrictions

Dim N As Long, i As Long, ss As String
Dim k1 As Long, k2 As Long, k3 As Long, k4 As Long
Dim salt As Boolean

salt = False

N = Len(s)
ss = Space(N)
ReDim sn(N) As Long

k1 = 11 + (key Mod 233): k2 = 7 + (key Mod 239)
k3 = 5 + (key Mod 241): k4 = 3 + (key Mod 251)

For i = 1 To N: sn(i) = Asc(Mid(s, i, 1)): Next

For i = 1 To N - 2: sn(i) = sn(i) Xor sn(i + 2) Xor (k4 * sn(i + 1)) Mod 256: Next
For i = N To 3 Step -1: sn(i) = sn(i) Xor sn(i - 2) Xor (k3 * sn(i - 1)) Mod 256: Next
For i = 1 To N - 1: sn(i) = sn(i) Xor sn(i + 1) Xor (k2 * sn(i + 1)) Mod 256: Next
For i = N To 2 Step -1: sn(i) = sn(i) Xor sn(i - 1) Xor (k1 * sn(i - 1)) Mod 256: Next

For i = 1 To N: Mid(ss, i, 1) = Chr(sn(i)): Next i

If salt Then StrDecode = Mid(ss, 3, Len(ss) - 4) Else StrDecode = ss

End Function

```

---

## hwb.exe

### Health & Well-Being utility

#### Overview

The **hwb** utility is an unattended database diagnostic and auto-maintenance utility used by the client to perform the following database procedures

1. check the database for tablespace fragmentation
1. check the tablespaces for available free space
1. check the hard drives for available free space
1. fix any problems that can be fixed automatically without risk

There is no user intervention required during the execution of hwb. All process messages and errors are written to a log file named hwb.log. The user is instructed to check this log each morning following a scheduled run of hwb. By default, hwb is scheduled to run once a week, on Sunday mornings at 11:00am. During the running of the Oracle sizing wizard (dbsizer) the user has the option to override this schedule.

Hwb's dialog box displays all the steps that it will perform during it's run. As each step is completed, a check mark will appear to the left of the step to signify it's completion. When all steps are complete, hwb will terminate automatically.

## Psedo-Code

Following is **pseudo-code** for the hwb utility program.

```
get the language setting from the NT Server registry
center the dialog
retrieve / decode and store Oracle database user ids and password from the registry
display the status dialog box
clear all the check marks next to each step
open the log file and note the start date and time
if not at least 1 MEG of free disk space on the \admin folder drive for scripts
    write an error to the log file
    exit
end if
shutdown the database (immediate mode)
restart the database in restricted mode
```

### ' step 1 begins (analyze tables, gather information)

```
coalesce all tablespaces
    run gencoal.sql which creates coalesce.sql
    run coalesce.sql
build a list (no_fix.out) of tables with > 1 extent but are too high risk to fix
generate no_fix.sql
    run no_fix.sql (creates no_fix.out)
if no_fix.out contains table names
    write a message to the log file and tell the user which tables need manual fixing
end if
run db_info.sql to generate report on database internals (db_info.txt, not used but
handy)
analyze tables
    generate bld_anal.sql
    if we have not analyzed tables today (stored in the registry)
        run bld_anal.sql which generates analyze.sql
        run analyze.sql
        store date in the registry so we don't do this again today
    end if
if xtra.sql exists in the \admin folder
    execute it (this allows us to implement one time procedures)
end if
build a list of all tables that can be fixed (fix_tab.out)
    generate fix_tab.sql
    run fix_tab.sql, which generates a list of tables that hwb should fix
display a check mark next to step 1
' step 1 complete
```

### ' step 2 begins (check database performance)

```
run perf.sql, generates perf.out which is a table of current performance
for each line written to perf.out
    lookup the performance criteria in the file perf.tbl
    if found
```

```

        compare database performance (perf.out) to error level (perf.tbl)
        if above error level
            write error to logfile
        else
            compare database performance (perf.out) to warning level
(perf.tbl)
            if above warning level
                write warning to logfile
            end if
        end if
    end if
    get next line from perf.out file
    make sure there's at least 5 MEG on each hard drive used to store PCPW data
    if any drive does not have at least 5 MEG free
        write message to log file
    end if
    display a check mark next to step 2
' step 2 complete

' step 3 fix low risk tables
' each step is stringently checked for errors and logged to the hwb.log file
open the fix_tab.out file which list tables to fix
for each line in the fix_tab.out file
    check each available drive to find one with enough disk space to hold export file
    if not
        write error to logfile
        skip this table, get the next line from fix_tab.out
    end if
    generate DDL script to rebuild primary key(s) (gen_pk.sql)
    generate DDL script to rebuild foreign key(s) (gen_fk.sql)
    export the data
    drop the table
    import the data from the export file
    run gen_pk.sql to rebuild primary key(s)
    run gen_fk.sql to rebuild foreign key(s)
    cleanup and get ready for the next table
get next line from fix_tab.out
display a check mark next to step 3
' step 3 complete

' step 4 rebuild indexes (if necessary)
run fix_idx.sql which generates rld_idx.sql
run rld_idx.sql to rebuild indexes if necessary
display a check mark next to step 4
' step 4 complete

cleanup any command files or script files left behind
note summary of warning and errors in the logfile (tally)
note completion date and time in the logfile
close the logfile
shutdown the database (immediate mode)
restart the database in normal mode
exit

```

## Command Line Parameters

The following command line parameters are recognized by the hwb utility

### **/DEBUG**

causes hwb to execute in *debug* mode. By default, hwb cleans up after itself deleting all temporary scripts and output files. When debugging, it is useful to look at these files so you can determine exactly what happened. **CAUTION:** this is extremely sensitive since SQL files and command files that contain the database password will be left on the hard drive in the \admin folder. Do not do this at a client site unless absolutely necessary, then when complete, **re-run the hwb utility WITHOUT the /debug flag to clean up the admin folder sufficiently!**

## NT Server - Registry Entries

When the Oracle sizing wizard is run by the client to create their database, a number of entries are written to the NT Server's system registry. The following entries are used by the hwb utility during execution

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWOr\Keys]
"PCPAYSYS"=" "
"INTERNAL"=" "
"MaintKey"=" "
"MIGRATE"="re_äY=Á "
"SUPEROP"=" "
"REPORTS"="uXy_^*f"
"Default"=" "
"SYS"=" "
"SYSTEM"="f3TNäYip"
```

These keys represent the user id's and passwords which can be part of a template (.brt) file. In order to use one of the user id / password combinations, the user id must be surrounded by %'s in the .brt file. For example, to use the SrvMgr23 utility to run a SQL file named dothis.sql and use the INTERNAL id and password, the following line would be in the dothis.brt file.

```
connect INTERNAL / %INTERNAL%
--some sql code here
```

At run time, hwb will retrieve the value for the INTERNAL key from the registry, decode the key value and write the following to the tempn.sql file in the c:\temp folder

```
connect INTERNAL / THEPASSWORD
--some sql code here
```

---

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWOr\Files]
"Home"="C:\\ORANT\\BIN"
"Maintenance"="C:\\ORADATA\\PCPW\\admin\\maint"
"Admin"="C:\\ORADATA\\PCPW\\ADMIN"
"Backup"=" "
```

These settings let hwb know where to find other files that it may need during execution

---

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWOr\Extents]
"Number"="1"
```

This settings tells hwb how many extents are acceptable. In this case, any tablespaces with more than 1 extent will be fixed.

---

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\PCPWOr\HWB]
"Tables"="1"
"Performance"="0"
"Use Note of the Day"="True"
```

These settings control some of the features of hwb. **Tables** tell hwb whether or not to check tablespaces during the database performance step. A 1 means Yes, a 0 means No. **Performance** tells hwb whether or not to check database engine performance criteria during the database performance step. **Use Note of the Day**. If "True" then fatal errors will generate a Note of the Day table entry. If "False" then fatal errors will **only** be logged to the hwb.log file. This is for client's who want to use the NT event log to monitor fatal errors. There is no way within the current version for hwb to write directly to the NT event log, but a client could write a program to analyze the hwb.log file and generate event entries. This is a good candidate for a PWR.

## Source Code

Following the source code for the hwb utility version 1.05-10.

```
VERSION 5.00
Begin VB.Form frmMain
    Caption           = "PCPW Health & Well Being Engine"
    ClientHeight      = 870
    ClientLeft        = 60
    ClientTop         = 345
    ClientWidth       = 5370
    LinkTopic         = "Form1"
    ScaleHeight       = 870
    ScaleWidth        = 5370
    StartUpPosition   = 2 'CenterScreen
End
Attribute VB_Name = "frmMain"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim g_szStatus As String
Dim g_dErrorCount As Integer
Dim g_szOracleHome As String
Dim g_szMaint As String

Private Sub Command1_Click()

    If g_szStatus = "READY" Then
        rc = WriteLogFile("HBW: Execute Stop")
        rc = CloseLogFile()
        End
    Else
        rc = MsgBox("Are you sure you want to cancel this process?", vbYesNo + vbQuestion,
"Confirm") = vbYes
        If rc = vbYes Then
            rc = WriteLogFile("HBW: Execute Stop")
            rc = CloseLogFile()
            End
        End If
    End If

End Sub

Private Sub Form_Load()

    Dim fh As Integer

    ' -----
    ' initialize
    ' -----
    Load frmStatus
    frmStatus.txtStatus.Text = "Ready."
    frmStatus.Show 0
    g_szStatus = "READY"
    g_dErrorCount = 0
    rc = OpenLogFile(App.Path & "\hbw")
    rc = WriteLogFile("HBW: Execute Start")

    fh = FreeFile
    Open App.Path & "\maint\fix.ctl" For Input As #fh
    Do Until EOF(fh)
```

```

        Line Input #fh, buf$
        If Mid$(buf$, 5, 8) = "ORA_HOME" Then
            g_szOracleHome = Trim(Mid$(buf$, 15))
        End If
        If Mid$(buf$, 5, 9) = "ORA_MAINT" Then
            g_szMaint = Trim(Mid$(buf$, 15))
        End If
    Loop

    If g_szOracleHome = "" Or g_szMaint = "" Then
        rc = WriteLogFile("HBW: *** ERROR *** Unable to open HBW config file (FIX.CTL)")
        rc = CloseLogFile()
        Unload frmStatus
        End
    End If

    Close #fh

    ' -----
    ' start fixfrag.sql
    ' -----
    frmStatus.txtStatus.Text = frmStatus.txtStatus.Text & Chr$(10) & "Checking your
database..."
    rc = WriteLogFile("HBW: Spawning FIXFRAG.SQL")
    rc = WriteLogFile("HBW: FIXFRAG.SQL return")

    ' -----
    ' analyze results -- check performance log
    ' -----
    frmStatus.txtStatus.Text = frmStatus.txtStatus.Text & Chr$(10) & "Checking performance
criteria..."
    rc = WriteLogFile("HBW: Checking Performance LOG for Warnings")
    rc = CheckPerf()

    ' -----
    ' analyze results -- check tbl/idx warnings
    ' -----
    frmStatus.txtStatus.Text = frmStatus.txtStatus.Text & Chr$(10) & "Checking
fragmentation..."
    rc = WriteLogFile("HBW: Checking Fragmentation Warnings")
    rc = CheckFragWarnings()

    ' -----
    ' analyze results -- check tbl/idx alarms
    ' -----
    rc = WriteLogFile("HBW: Checking Fragmentation Alarms")
    rc = CheckFragAlarms()

    ' -----
    ' all done, close up shop
    ' -----
    frmStatus.txtStatus.Text = frmStatus.txtStatus.Text & Chr$(10) & "Process complete,
cleaning up..."
    rc = WriteLogFile("HBW: Process reached completion")

    If g_dErrorCount = 0 Then
        rc = WriteLogFile("HBW: There were no errors reported.")
    Else
        rc = WriteLogFile("HBW: *** " & g_dErrorCount & " *** Errors reported.")
    End If

    rc = CloseLogFile()
    Unload frmStatus
    End

End Sub

```

```

Private Function CheckPerf() As Boolean

    Dim fh As Integer

    ' -----
    ' open performance log file
    ' -----

    fh = FreeFile
    On Error GoTo NoPerfLog
    Open App.Path & "..\maint\perf.log" For Input As #fh

    ' -----
    ' if any warnings, write NOTE OF THE DAY ENTRY
    ' and record in HBW log file
    ' -----
    Do Until EOF(fh)

    Loop

    ' -----
    ' close performance log file
    ' -----
    Close #fh

    On Error GoTo 0
    CheckPerf = True
    rc = WriteLogFile("HBW:      Performance analysis complete.")
    Exit Function

NoPerfLog:
    On Error GoTo 0
    rc = WriteLogFile("HBW:      ** ERROR ** Unable to open Performance Log File.
(PERF.LOG)")
    g_dErrorCount = g_dErrorCount + 1
    Exit Function

End Function

Private Function CheckFragWarnings() As Boolean

    Dim fh As Integer

    ' -----
    ' open frag warning log file
    ' -----

    fh = FreeFile
    On Error GoTo NoWarnLog
    Open App.Path & "..\maint\fragwarn.log" For Input As #fh

    ' -----
    ' if any entries, write NOTE OF THE DAY ENTRY
    ' and record in HBW log file
    ' -----
    Do Until EOF(fh)

    Loop

    ' -----
    ' close frag warning log file
    ' -----
    Close #fh

    On Error GoTo 0
    CheckFragWarnings = True
    rc = WriteLogFile("HBW:      Fragmentation analysis complete.")
    Exit Function

```

```

NoWarnLog:
    On Error GoTo 0
    rc = WriteLogFile("HBW:      ** ERROR ** Unable to open Fragmentation Log File.
(FRAGWARN.LOG)")
    g_dErrorCount = g_dErrorCount + 1
    CheckFragWarnings = False
    Exit Function

End Function

Private Function CheckFragAlarms() As Boolean

    Dim fh As Integer
    Dim fh2 As Integer
    Dim buf As String
    Dim tName As String
    Dim tSize As Double

    ' -----
    ' open frag alarm log file
    ' -----

    fh = FreeFile
    On Error GoTo NoAlarmLog
    Open App.Path & "\maint\fix_tab.out" For Input As #fh

    ' -----
    ' if any entries, FIX 'EM
    ' -----

    If LOF(fh) > 0 Then

        ' -----
        ' shutdown the database and bring it back
        ' up in restricted mode
        ' -----

        frmStatus.txtStatus.Text = frmStatus.txtStatus.Text & Chr$(10) & "Fixing
fragmentation..."

        ' -----
        ' for each entry in the alarm log file
        ' -----

        Do Until EOF(fh)

            Line Input #fh, buf

            ' -----
            ' get the table name and the required disk space
            ' for the export file
            ' -----

            tName = Trim(Left$(buf, 30))
            tSize = Val(Trim(Mid$(buf, 31, 20)))
            rc = WriteLogFile("HBW:      Fixing " & tName & "(" & Format$(tSize, "###0") &
" bytes required)")

            ' -----
            ' find a drive that can handle it
            ' -----

            szDrive$ = FindSpace(tSize, "C:")
            If szDrive$ = "" Then
                rc = WriteLogFile("HBW:      *** ERROR *** Can't find drive with " &
Format$(tSize, "###0") & " bytes free. Cannot create export file.")
                CheckFragAlarms = False
                g_dErrorCount = g_dErrorCount + 1
                Exit Function
            Else
                rc = WriteLogFile("HBW:      Export file (" & tName & ".dmp) will be

```

```

created on " & szDrive)
End If

' -----
' generate DDL
' -----
' Primary Key
' -----

fh2 = FreeFile
Open App.Path & "\maint\gen_pk.sql" For Output As #fh2
Print #fh2, "connect pcpayssys/pay4946"
Print #fh2, "spool " & App.Path & "\maint\drop.log"
Print #fh2, "SELECT 'ALTER TABLE ' || UPPER('" & tName & "') || ' ADD (PRIMARY
KEY (' || column_name"
Print #fh2, "From"
Print #fh2, "    user_cons_columns T1,"
Print #fh2, "    user_constraints T2"
Print #fh2, "Where"
Print #fh2, "    T1.table_name = UPPER('" & tName & "')"
Print #fh2, "    AND constraint_type = 'P'"
Print #fh2, "    AND T1.constraint_name = T2.constraint_name"
Print #fh2, "    AND position = 1"
Print #fh2, "/"
Print #fh2, "SELECT"
Print #fh2, "    ',' || column_name"
Print #fh2, "From"
Print #fh2, "    user_cons_columns T1,"
Print #fh2, "    user_constraints T2"
Print #fh2, "Where"
Print #fh2, "    T1.table_name = UPPER('" & tName & "')"
Print #fh2, "    AND constraint_type = 'P'"
Print #fh2, "    AND T1.constraint_name = T2.constraint_name"
Print #fh2, "    AND position > 1"
Print #fh2, "Order By"
Print #fh2, "    position"
Print #fh2, "/"
Print #fh2, "SELECT '));'"
Print #fh2, "From DUAL"
Print #fh2, "/"
Close #fh2

fh2 = FreeFile
Open App.Path & "\maint\gen_pk.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\gen_pk.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\gen_pk.bat")
' TODO: check for success
rc = WriteLogFile("HBW:      DDL generation for " & tName & "(PK) complete.")

' -----
' Foreign Key(s)
' -----

fh2 = FreeFile
Open App.Path & "\maint\gen_fk1.sql" For Output As #fh2
Print #fh2, "spool " & App.Path & "\maint\fk1.sql"
Print #fh2, "SELECT 'spool " & App.Path & "\maint\fk.sql' from dual"
Print #fh2, "/"
Print #fh2, ""
Print #fh2, "/* Generate all Parent Foreign Keys */"
Print #fh2, ""
Print #fh2, "SELECT '@' & App.Path & "\maint\Gen_fk2.sql ' || UPPER('" & tName
& "') || ' ' || constraint_name"
Print #fh2, "From"
Print #fh2, "user_constraints"
Print #fh2, "Where"
Print #fh2, "table_name = UPPER('" & tName & "')"

```

```

Print #fh2, "AND      constraint_type = 'R'"
Print #fh2, "ORDER BY constraint_name"
Print #fh2, "/"
Print #fh2, ""
Print #fh2, "/* Generate all Children Foreign Keys */"
Print #fh2, ""
Print #fh2, "SELECT '@' & App.Path & "\maint\Gen_fk2.sql ' || t1.table_name ||
' ' || t1.constraint_name"
Print #fh2, "From"
Print #fh2, "user_constraints T1,"
Print #fh2, "user_constraints t2"
Print #fh2, "Where"
Print #fh2, "t2.table_name = '" & tName & "' and t2.constraint_type = 'P'"
Print #fh2, "AND t2.constraint_name = t1.r_constraint_name"
Print #fh2, "ORDER BY t1.table_name"
Print #fh2, "/"
Print #fh2, ""
Print #fh2, "SELECT 'spool off' FROM dual"
Print #fh2, "/"
Print #fh2, ""
Print #fh2, "spool off"
Close #fh2

fh2 = FreeFile
Open App.Path & "\maint\gen_fk1.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\gen_fk1.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\gen_fk1.bat")
' TODO: check for success
rc = WriteLogFile("HBW:      DDL generation for " & tName & "(FK1)  complete.")

fh2 = FreeFile
Open App.Path & "\maint\gen_fk1.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\fk1.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\gen_fk1.bat")
' TODO: check for success
rc = WriteLogFile("HBW:      DDL generation for " & tName & "(FK)  complete.")

' -----
' do the export
' -----
' create exp.sql in the maint folder and execute it using
' SVRMGR23 (NT), the export statement looks like...
' c:\orant\bin\exp73 username=pcpaysys/pay4946 constraints=n
tables=(t_schedule) file=d:\export\t_schedule.dmp
log=c:\oradata\pcpw\maint\t_schedule_exp.log
fh2 = FreeFile
Open App.Path & "\maint\export.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\exp73 username=pcpaysys/pay4946 constraints=n
tables=(" & tName & ") file=" & szDrive$ & "\export\" & tName & ".dmp log=" & g_szMaint &
"\ " & tName & "_exp.log"
Close #fh2

' now, execute the bat file just created in the step above
bDir = False
If Dir$(szDrive$ & "\export", vbDirectory) = "" Then
    MkDir szDrive$ & "\export"
    bDir = True
End If
ExecDOSCmd (App.Path & "\maint\export.bat")
' TODO: check for success
rc = WriteLogFile("HBW:      Export complete.")
' -----

```

```

' drop the table
' -----
fh2 = FreeFile
Open App.Path & "\maint\drop.sql" For Output As #fh2
Print #fh2, "connect pcpayssys/pay4946"
Print #fh2, "spool '" & App.Path & "\maint\drop.log'"
Print #fh2, "DROP TABLE " & tName & " CASCADE CONSTRAINTS;"
Close #fh2

fh2 = FreeFile
Open App.Path & "\maint\drop.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\drop.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\drop.bat")
' TODO: check drop.log for success
rc = WriteLogFile("HBW:      Table " & tName & " dropped.")

' -----
' import the exported data
' -----
fh2 = FreeFile
Open App.Path & "\maint\import.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\imp73 username=pcpayssys/pay4946 constraints=n
tables=(" & tName & ") file=" & szDrive$ & "\export\" & tName & ".dmp log=" & g_szMaint &
"\\" & tName & "_imp.log"
Close #fh2

ExecDOSCmd (App.Path & "\maint\import.bat")
' TODO: check for success
rc = WriteLogFile("HBW:      Import complete.")

' -----
' use generated DLL to recreate constraints
' -----
fh2 = FreeFile
Open App.Path & "\maint\ddl.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\pk.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\ddl.bat")
' TODO: check drop.log for success
rc = WriteLogFile("HBW:      Primary key created.")

fh2 = FreeFile
Open App.Path & "\maint\ddl.bat" For Output As #fh2
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\fk.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\ddl.bat")
' TODO: check drop.log for success
rc = WriteLogFile("HBW:      Foreign key(s) created.")

' -----
' cleanup and get ready for the next table
' -----
If Dir$(szDrive$ & "\export\" & tName & ".dmp", vbNormal) <> "" Then
    Kill szDrive$ & "\export\" & tName & ".dmp"
End If
If bDir Then
    Rmdir szDrive$ & "\export"
End If

' TODO: clean up maint folder

Loop

```

```

' -----
' all entries processed, so shutdown the database
' and bring it back up in normal mode.
' -----

rc = WriteLogFile("HBW:      Fragmentation repairs complete.")

Else

rc = WriteLogFile("HBW:      No fragmentation repairs necessary.")

End If

' -----
' close frag alarm log file
' -----

Close #fh
On Error GoTo 0
CheckFragAlarms = True
Exit Function

NoAlarmLog:
On Error GoTo 0
rc = WriteLogFile("HBW:      ** ERROR ** Unable to open Fragmentation Fix Log File.
(FIX_TAB.OUT) ")
g_dErrorCount = g_dErrorCount + 1
CheckFragAlarms = False
Exit Function

End Function

Function FindSpace(spaceNeeded As Double, startingDrive As String) As String

Dim di As New clsDiskInfo
Dim freebytes As Double

' -----
' see if fn can fit on fdr ( size is ns )
' -----
freebytes = GetDiskFreeSpaceLarge(startingDrive)
If freebytes > spaceNeeded Then
' -----
' it fits, so just put it here
' -----
FindSpace = startingDrive
Exit Function
End If

' -----
' doesn't fit, so check other drives
' -----
dbFound = False
For i = 1 To 26

If di.DriveType(Chr$(64 + i)) = 3 Or di.DriveType(Chr$(64 + i)) = 4 Then
di.PathName = Chr$(64 + i) + ":\\"
freebytes = GetDiskFreeSpaceLarge(di.PathName)
' -----
' adjust freebytes for any dbfs that are already
' targetted for this drive
' -----
If freebytes > spaceNeeded Then
' -----
' it fits here, so put it here
' -----
dbFound = True

```

```

        Exit Function
    End If
End If

Next i

If dbFound = False Then
    FindSpace = ""
Else
    FindSpace = di.PathName
End If

End Function

Public Function GetDiskFreeSpaceLarge(DriveLetter As String) As Double

    Dim hdb As Integer
    Dim bf As Double
    Dim buf As String
    Dim buf2 As String
    Dim fh As Integer

    fh = FreeFile
    On Error GoTo CantWrite
    Open DriveLetter + "test.txt" For Output As #fh
    On Error GoTo 0
    Print #fh, "Testing"
    Close #fh

    ExecDOSCmd ("command.com /c dir " + DriveLetter + "test.txt > c:\dbsizer.lst")
    ExecDOSCmd ("cmd /c dir " + DriveLetter + "test.txt > c:\dbsizer.lst")
    hdb = FreeFile
    Open "c:\dbsizer.lst" For Input As #hdb
    Do Until EOF(hdb)
        Line Input #hdb, buf
        idx = InStr(buf, "bytes free")
        If idx > 0 Then
            buf = Left$(buf, idx - 2)
            For i% = Len(buf) To 1 Step -1
                If Mid$(buf, i%, 1) = " " Then
                    buf = Mid$(buf, i% + 1)
                Exit For
            End If
        Next i%
    End If
    Loop
    Close #hdb
    Kill "c:\dbsizer.lst"
    Kill DriveLetter + "test.txt"

    buf2 = ""
    For i% = 1 To Len(buf)
        thisChar = Mid$(buf, i%, 1)
        If thisChar <> " " And thisChar <> "," Then
            buf2 = buf2 + thisChar
        End If
    Next i%

    GetDiskFreeSpaceLarge = Val(buf2)
    Exit Function

CantWrite:
    On Error GoTo 0
    GetDiskFreeSpaceLarge = 0

End Function

```

VERSION 5.00

Begin VB.Form frmStatus

```
BackColor      = &H00FFFFFF&
Caption        = "PCPW/Oracle Health Check"
ClientHeight   = 3165
ClientLeft     = 60
ClientTop      = 345
ClientWidth    = 6900
Icon           = "frmStatus.frx":0000
LinkTopic      = "Form1"
ScaleHeight    = 3165
ScaleWidth     = 6900
StartPosition  = 2 'CenterScreen
```

Begin VB.CommandButton Command1

```
Caption        = "Cancel"
Height         = 360
Left           = 5685
TabIndex       = 14
Top            = 2700
Visible        = 0 'False
Width          = 1080
```

End

Begin VB.PictureBox Picture3

```
Appearance     = 0 'Flat
BackColor      = &H000000FF&
BorderStyle    = 0 'None
ForeColor      = &H80000008&
Height         = 645
Left           = -15
ScaleHeight    = 645
ScaleWidth     = 6915
TabIndex       = 12
Top            = 0
Width          = 6915
```

Begin VB.Label Label1

```
Alignment      = 2 'Center
BackStyle      = 0 'Transparent
Caption        = "Do not interrupt! Database maintenance in progress..."
```

BeginProperty Font

```
Name          = "Arial"
Size           = 12
Charset        = 0
Weight         = 700
Underline      = 0 'False
Italic         = 0 'False
Strikethrough  = 0 'False
```

EndProperty

```
ForeColor      = &H00FFFFFF&
Height         = 315
Left           = 165
TabIndex       = 13
Top            = 150
Width          = 6540
```

End

End

Begin VB.PictureBox Picture2

```
AutoSize       = -1 'True
BorderStyle    = 0 'None
Height         = 1245
Left           = 180
Picture        = "frmStatus.frx":1CFA
ScaleHeight    = 1245
ScaleWidth     = 2250
TabIndex       = 10
Top            = 855
Width          = 2250
```

```

End
Begin VB.PictureBox Picture1
    Appearance      = 0 'Flat
    AutoRedraw      = -1 'True
    AutoSize        = -1 'True
    BackColor       = &H80000005&
    BorderStyle     = 0 'None
    ForeColor       = &H80000008&
    Height          = 225
    Index           = 4
    Left            = 2505
    Picture         = "frmStatus.frx":AFC8
    ScaleHeight     = 225
    ScaleWidth      = 240
    TabIndex        = 9
    Top             = 2280
    Width           = 240
End
Begin VB.PictureBox Picture1
    Appearance      = 0 'Flat
    AutoRedraw      = -1 'True
    AutoSize        = -1 'True
    BackColor       = &H80000005&
    BorderStyle     = 0 'None
    ForeColor       = &H80000008&
    Height          = 225
    Index           = 3
    Left            = 2505
    Picture         = "frmStatus.frx":B2DA
    ScaleHeight     = 225
    ScaleWidth      = 240
    TabIndex        = 8
    Top             = 1920
    Width           = 240
End
Begin VB.PictureBox Picture1
    Appearance      = 0 'Flat
    AutoRedraw      = -1 'True
    AutoSize        = -1 'True
    BackColor       = &H80000005&
    BorderStyle     = 0 'None
    ForeColor       = &H80000008&
    Height          = 225
    Index           = 2
    Left            = 2505
    Picture         = "frmStatus.frx":B5EC
    ScaleHeight     = 225
    ScaleWidth      = 240
    TabIndex        = 7
    Top             = 1560
    Width           = 240
End
Begin VB.PictureBox Picture1
    Appearance      = 0 'Flat
    AutoRedraw      = -1 'True
    AutoSize        = -1 'True
    BackColor       = &H80000005&
    BorderStyle     = 0 'None
    ForeColor       = &H80000008&
    Height          = 225
    Index           = 1
    Left            = 2505
    Picture         = "frmStatus.frx":B8FE
    ScaleHeight     = 225
    ScaleWidth      = 240
    TabIndex        = 6
    Top             = 1200

```

```

        Width          = 240
End
Begin VB.PictureBox Picture1
    Appearance          = 0 'Flat
    AutoRedraw          = -1 'True
    AutoSize            = -1 'True
    BackColor           = &H80000005&
    BorderStyle         = 0 'None
    ForeColor           = &H80000008&
    Height              = 225
    Index               = 0
    Left                = 2505
    Picture             = "frmStatus.frx":BC10
    ScaleHeight         = 225
    ScaleWidth          = 240
    TabIndex            = 5
    Top                 = 840
    Width               = 240
End
Begin VB.Label lblTicker
    BackStyle           = 0 'Transparent
    Caption             = "Label3"
    Height              = 240
    Left                = 135
    TabIndex            = 11
    Top                 = 2790
    Width               = 5490
End
Begin VB.Label Label2
    BackStyle           = 0 'Transparent
    Caption             = "Rebuilding indexes"
    Height              = 255
    Index               = 4
    Left                = 2835
    TabIndex            = 4
    Top                 = 2310
    Width               = 4005
End
Begin VB.Label Label2
    BackStyle           = 0 'Transparent
    Caption             = "Checking index fragmentation"
    Height              = 255
    Index               = 3
    Left                = 2835
    TabIndex            = 3
    Top                 = 1950
    Width               = 4005
End
Begin VB.Label Label2
    BackStyle           = 0 'Transparent
    Caption             = "Checking table fragmentation"
    Height              = 255
    Index               = 2
    Left                = 2835
    TabIndex            = 2
    Top                 = 1590
    Width               = 4035
End
Begin VB.Label Label2
    BackStyle           = 0 'Transparent
    Caption             = "Checking database performance statistics"
    Height              = 255
    Index               = 1
    Left                = 2835
    TabIndex            = 1
    Top                 = 1230
    Width               = 3990

```

```

End
Begin VB.Label Label2
    BackStyle      = 0 'Transparent
    Caption        = "Analyzing database"
    Height         = 255
    Index          = 0
    Left           = 2835
    TabIndex       = 0
    Top            = 870
    Width          = 4005
End
End
Attribute VB_Name = "frmStatus"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Command1_Click()

    g_Cancel = True

End Sub

Private Sub Form_Load()

    Dim i As Integer

    ' -----
    ' get the language
    ' -----
    g_LANGUAGE = RegGetValue(HKEY_CURRENT_USER, "Control Panel\International", "Locale")

    Select Case g_LANGUAGE
        Case "00001009"
            g_LANGOFFSET = 1000
            g_VARTABLE = "Variables" & g_LANGUAGE
        Case "00000C0C"
            g_LANGOFFSET = 2000
            g_VARTABLE = "Variables" & g_LANGUAGE
        Case Else
            g_LANGOFFSET = 0
            g_VARTABLE = "Variables00000409"
    End Select

    On Error GoTo NoLanguageRes

    txt$ = RES(101)
    GoTo LanguageContinue

NoLanguageRes:
    g_LANGOFFSET = 0
    g_VARTABLE = "Variables00000409"

LanguageContinue:
    On Error GoTo 0

    ' -----
    ' DEBUG: uncomment the next line to force language selection
    ' -----
    ' g_LANGOFFSET = 2000
    ' g_VARTABLE = "Variables" & g_LANGUAGE

    i = SetWindowPos(Me.hWnd, HWND_TOPMOST, _
        Me.Left \ Screen.TwipsPerPixelX, Me.Top \ Screen.TwipsPerPixelY, _
        Me.Width \ Screen.TwipsPerPixelX, Me.Height \ Screen.TwipsPerPixelY, 0)

    Me.Caption = RES(101)

```

```

Label1.Caption = RES(203)
Label2(0).Caption = RES(204)
Label2(1).Caption = RES(205)
Label2(2).Caption = RES(206)
Label2(3).Caption = RES(207)
Label2(4).Caption = RES(208)
Command1.Caption = RES(209)

End Sub

Attribute VB_Name = "vbLogFile"
Dim m_dFH As Integer
Dim m_szLogFile As String
Dim m_lCount As Long

Public Function OpenLogFile(szApp As String, szDate As Date) As Boolean
' -----
' this function will open a log file for the specified
' application (szApp). the format of the log file will be
' appname.log. If a date is passed in the second parameter
' then the log file will be trimmed using the TrimLog function
' -----
m_szLogFile = GetParam(szApp, 1, ".", True)
If szDate <> 0 Then
    rc = TrimLog(m_szLogFile, szDate)
End If
m_szLogFile = m_szLogFile & ".log"

m_dFH = FreeFile
Open m_szLogFile For Append As #m_dFH
If LOF(m_dFH) = 0 Then
    rc = WriteLogFile(UCASE$(m_szLogFile))
    rc = WriteLogFile(Format$(Now, "yyyy-mm-dd hh:mm:ss -- "))
End If
rc = WriteLogFile("DELIMITER")
OpenLogFile = True
End Function

Public Function CloseLogFile() As Boolean
Close #m_dFH
CloseLogFile = False
End Function

Public Function WriteLogFile(szText As String) As Boolean
If szText = "DELIMITER" Then
    Print #m_dFH, Format$(Now, "yyyy-mm-dd hh:mm:ss -- ") &
"*****"
Else
    Print #m_dFH, Format$(Now, "yyyy-mm-dd hh:mm:ss -- ") & szText
End If
WriteLogFile = True
End Function

Private Function TrimLog(szLogFile As String, szDate As Date) As Boolean
' -----
' this function will trim the oldest messages from a
' log file. The parameter szDate specifies the date of
' the oldest message to keep. Any message dated before the
' specified date will be deleted.
' -----
Dim fh, fh2 As Integer
Dim tDate As Date
Dim szFile As String

szFile = szLogFile & ".log"
If Dir$(szFile, vbNormal) = "" Then
    TrimLog = True

```

```

        Exit Function
    End If
    If FileLen(szFile) = 0 Then
        TrimLog = True
        Exit Function
    End If

    fh = FreeFile
    Open szLogFile & ".log" For Input As #fh

    fh2 = FreeFile
    Open szLogFile & ".tmp" For Output As #fh2

    ' first line in a log file is always the original creation date
    Line Input #fh, t$
    Print #fh2, t$
    ' get the second line which is always the last trim date
    Line Input #fh, t$
    Print #fh2, Format$(Now, "YYYY-MM-DD hh:mm:ss -- ") & RES(202) & szDate

    Do Until EOF(fh)
        Line Input #fh, t$
        tDate = CDate(Left$(t$, 19))
        If tDate >= szDate Then
            Print #fh2, t$
        End If
    Loop

    Close #fh
    Close #fh2

    FileCopy szLogFile & ".tmp", szLogFile & ".log"
    Kill szLogFile & ".tmp"
    TrimLog = True

End Function

Attribute VB_Name = "Module1"
Public Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, _
    ByVal hWndInsertAfter As Integer, _
    ByVal X As Integer, ByVal Y As Integer, _
    ByVal cx As Integer, ByVal cy As Integer, _
    ByVal wFlags As Integer) As Integer

Global g_LANGUAGE As String
Global g_LANGOFFSET As Integer
Global g_VARTABLE As String

Global Const SWP_NOMOVE = 2
Global Const SWP_NOSIZE = 1
Global Const WndFlags = SWP_NOMOVE Or SWP_NOSIZE
Global Const HWND_TOPMOST = -1
Global Const HWND_NOTOPMOST = -2

Global g_DEBUG_MODE As Boolean
Global g_HWBTables As Integer
Global g_HWBPerf As Integer
Global g_MEGFree As Double
Global g_HWBLASTANALYZE As Date

Type bufLayout
    ItemType As String * 1
    ItemName As String * 30
    CompareType As String * 1
    WarningValue As String * 5
    ErrorValue As String * 5
    ActualValue As String * 5

```

```

End Type

Type PerfStats
    ItemType As String * 1
    ItemName As String * 30
    CompareType As String * 1
    WarningValue As String * 5
    ErrorValue As String * 5
    ActualValue As String * 5
    Count As Integer
End Type

Dim g_szStatus As String
Dim g_dErrorCount As Integer
Dim g_dWarningCount As Integer
Dim g_szOracleHome As String
Dim g_szMaint As String
Dim g_Password As String
Dim g_MaintPassword As String
Dim g_dLogFileAge As Integer
Dim g_dExtents As Integer
Dim g_WriteNoteoftheDay As String

Sub Main()

    Dim fh As Integer
    Dim m_Date As Date

    If UCase$(Command$) = "/DEBUG" Then
        g_DEBUG_MODE = True
    Else
        g_DEBUG_MDOE = False
    End If

    ' get passwords from System Registry and decrypt tem...
    g_Password = GetSetting("PCPWora", "Keys", "INTERNAL", "")
    g_Password = StrDecode(g_Password, 14755)

    g_MaintPassword = GetSetting("PCPWora", "Keys", "PCPAYSYS", "")
    g_MaintPassword = StrDecode(g_MaintPassword, 14755)

    ' get other processing parameters from Registry
    g_dLogFileAge = Val(GetSetting("PCPWora", "LogFiles", "Age", "90"))
    g_dExtents = Val(GetSetting("PCPWora", "Extents", "Number", "1"))
    g_HWBTables = Val(GetSetting("PCPWora", "HWB", "Tables", "1"))
    g_HWBPerf = Val(GetSetting("PCPWora", "HWB", "Performance", "1"))
    g_MEGFree = Val(GetSetting("PCPWora", "HWB", "Disk Space Warning", "5"))
    g_HWBLastANALYZE = CDate(GetSetting("PCPWora", "HWB", "Last Analyze", "01/01/80"))
    g_WriteNoteoftheDay = GetSetting("PCPWora", "HWB", "Use Note of the Day", "True")

    Load frmStatus
    For i = 0 To 4
        frmStatus.Picture1(i).Visible = False
        frmStatus.Label2(i).FontBold = False
    Next i
    frmStatus.Label2(0).FontBold = True
    frmStatus.lblTicker.Caption = ""
    frmStatus.Show 0
    frmStatus.Refresh

    ' -----
    ' initialize
    ' -----

    g_dErrorCount = 0
    g_dWarningCount = 0

```

```

m_Date = Now - g_dLogFileAge
rc = OpenLogFile(App.Path & "\hwb", m_Date)
rc = WriteLogFile("HBW: Execute Start")
g_szOracleHome = GetSetting("PCPWOr", "Files", "Home", "")
g_szMaint = GetSetting("PCPWOr", "Files", "Maintenance", "")

On Error GoTo NoLogsToDelete
Kill App.Path & "\maint\*.log"
NoLogsToDelete:
On Error GoTo 0

' make sure there's at least 1 MEG of free space on the admin drive
cDrive$ = Mid$(GetSetting("PCPWOr", "Files", "Admin", ""), 1, 1) & ":\\"
freebytes = GetDiskFreeSpaceLarge(cDrive$) / 1024000
If freebytes < 1 Then
    rc = WriteLogFile(RES(102)) ' not enough disk space
    Call CleanUp
    rc = CloseLogFile()
    Unload frmStatus
    End
End If

' -----
' shutdown the database and bring it back
' up in restricted mode
' -----
fh = FreeFile
Open App.Path & "\maint\shutdown.sql" For Output As #fh
Print #fh, "connect internal/" & g_Password
Print #fh, "shutdown immediate"
Close #fh

fh = FreeFile
Open App.Path & "\maint\shutdown.bat" For Output As #fh
Print #fh, "set ORACLE_SID=PCPW"
Print #fh, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\shutdown.sql"
Close #fh

frmStatus.lblTicker.Caption = RES(103) ' shutting down the database
frmStatus.Refresh
DoEvents

' -----
' Before shutting down, get the length of the alert log
' so I don't have to read the whole thing to get to the
' end
' -----
lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")
ExecDOSCmd (App.Path & "\maint\shutdown.bat")

' -----
' now that the database is shutdown, make sure the shutdown
' was successful and without errors
' -----
fhCheck = FreeFile
Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
Seek #fhCheck, lAlertLogLength
bClosed = False
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$
    If Trim(buf$) = "Completed: ALTER DATABASE DISMOUNT" Then
        bClosed = True
    End If
    If Trim(buf$) = "Completed: ALTER DATABASE pay4win DISMOUNT" Then
        bClosed = True
    End If
    DoEvents

```

```

    Loop
Close #fhCheck
If bClosed = False Then
    ' -----
    ' this means the database was not shutdown properly
    ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
    ' table, then get out.
    ' -----
    rc = WriteLogFile(RES(104)) ' unable to shutdown
    rc = WriteNoteOfTheDay(RES(105)) ' unable to shutdown
    BailOut (False)
End If

fh = FreeFile
Open App.Path & "\maint\restrict.sql" For Output As #fh
Print #fh, "connect internal/" & g_Password
Print #fh, "startup pfile=" & App.Path & "\initpcpw.ora restrict"
Close #fh

fh = FreeFile
Open App.Path & "\maint\restrict.bat" For Output As #fh
Print #fh, "set ORACLE_SID=PCPW"
Print #fh, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\restrict.sql"
Close #fh

frmStatus.lblTicker.Caption = RES(106) ' starting in restricted mode
frmStatus.Refresh
DoEvents

lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")
ExecDOSCmd (App.Path & "\maint\restrict.bat")
' -----
' Make sure the database is up in restricted mode
' -----
fhCheck = FreeFile
Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
Seek #fhCheck, lAlertLogLength
bClosed = False
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$
    If Trim(buf$) = "Completed: alter database open" Then
        bClosed = True
    End If
    If Trim(buf$) = "Completed: alter database pay4win open" Then
        bClosed = True
    End If
    DoEvents
Loop
Close #fhCheck
If bClosed = False Then
    ' -----
    ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
    ' table, then get out.
    ' -----
    rc = WriteLogFile(RES(107)) ' unable to restart
    rc = WriteNoteOfTheDay(RES(108)) ' unable to restart
    BailOut (True)
End If

' -----
' before we begin, coalesce all tablespaces
' -----
If Dir$(App.Path & "\gencoal.sql", vbNormal) = "" Then
    rc = WriteLogFile(RES(109)) ' control file missing
    WriteNoteOfTheDay (RES(110)) ' alert admin
    BailOut (True)
End If

```

```

    fh = FreeFile
    Open App.Path & "\gencoal.bat" For Output As #fh
    Print #fh, "set ORACLE_SID=PCPW"
    Print #fh, g_szOracleHome & "\sqlplus pcpsys/" & g_MaintPassword & " @" & App.Path &
"\gencoal.sql " & App.Path
    Close #fh

    ' clear the flag file
    rc = RemoveFile(App.Path & "\gcdone.out")
    ExecDOSCmd (App.Path & "\gencoal.bat")

    frmStatus.lblTicker.Caption = RES(111) ' coalescing tables. pass 1
    ' make sure the prior step is complete before continuing
    Do Until Dir$(App.Path & "\gcdone.out", vbNormal) <> ""
        DoEvents
    Loop

    fh = FreeFile
    Open App.Path & "\coalesce.bat" For Output As #fh
    Print #fh, "set ORACLE_SID=PCPW"
    Print #fh, g_szOracleHome & "\sqlplus pcpsys/" & g_MaintPassword & " @" & App.Path &
"\coalesce.sql " & App.Path & "\"
    Close #fh

    ' clear the flag file
    rc = RemoveFile(App.Path & "\coaldone.out")
    ExecDOSCmd (App.Path & "\coalesce.bat")

    frmStatus.lblTicker.Caption = RES(112) ' coalescing tables. pass 2
    ' make sure the prior step is complete before continuing
    Do Until Dir$(App.Path & "\coaldone.out", vbNormal) <> ""
        DoEvents
    Loop

    rc = RemoveFile(App.Path & "\gencoal.bat")
    rc = RemoveFile(App.Path & "\coalesce.bat")

    ' -----
    ' start fix_tab.sql
    ' -----
    ' if unable to run fix_tab.sql then put a note of the day
    ' and bail out.
    ' -----
    rc = WriteLogFile(RES(113)) ' checking Table/Index fragmentation
    rc = FixTables()
    frmStatus.Picture1(0).Visible = True
    frmStatus.Label2(0).FontBold = False
    frmStatus.Label2(1).FontBold = True
    frmStatus.Refresh

    ' -----
    ' analyze results -- check performance log
    ' -----
    rc = WriteLogFile(RES(114)) ' checking/fixing performance criteria
    rc = CheckPerf()
    frmStatus.Picture1(1).Visible = True
    frmStatus.Label2(1).FontBold = False
    frmStatus.Label2(2).FontBold = True
    frmStatus.Refresh

    ' -----
    ' analyze results -- check tbl/idx alarms
    ' -----
    rc = WriteLogFile(RES(115)) ' HBW: Fixing Table/Index Fragmentation
    rc = CheckFragAlarms()
    frmStatus.Picture1(2).Visible = True

```

```

frmStatus.Label2(2).FontBold = False
frmStatus.Label2(3).FontBold = True
frmStatus.Refresh

frmStatus.Picture1(3).Visible = True
frmStatus.Label2(3).FontBold = False
frmStatus.Label2(4).FontBold = True
frmStatus.Refresh

frmStatus.Picture1(4).Visible = True
frmStatus.Label2(4).FontBold = False
frmStatus.Refresh

' -----
' all done, close up shop, and write Note of the Day
' messages if necessary
' -----
rc = WriteLogFile(RES(116)) ' HBW: Process reached completion

If g_dErrorCount = 0 Then
    If g_dWarningCount = 0 Then
        rc = WriteLogFile(RES(117)) ' HBW: There were no warnings reported.
        rc = WriteLogFile(RES(118)) ' HBW: There were no errors reported.
        rc = ClearNoteOfTheDay(RES(110))
        rc = ClearNoteOfTheDay("")
    Else
        rc = WriteLogFile(RES(119) & g_dWarningCount & RES(120))
        rc = WriteLogFile(RES(118))
        rc = WriteNoteOfTheDay(RES(110))
    End If
Else
    If g_dWarningCount = 0 Then
        rc = WriteLogFile(RES(117))
        rc = WriteLogFile(RES(119) & g_dErrorCount & RES(121))
        rc = WriteNoteOfTheDay("")
    Else
        rc = WriteLogFile(RES(119) & g_dWarningCount & RES(120))
        rc = WriteLogFile(RES(119) & g_dErrorCount & RES(121))
        rc = WriteNoteOfTheDay("")
    End If
End If

' -----
' all entries processed, so shutdown the database
' and bring it back up in normal mode.
' -----
frmStatus.lblTicker.Caption = RES(103) ' shutting down the database
frmStatus.Refresh
DoEvents

lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")
ExecDOSCmd (App.Path & "\maint\shutdown.bat")
' -----
' Make sure the database is shutdown
' -----
fhCheck = FreeFile
Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
Seek #fhCheck, lAlertLogLength
bClosed = False
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$
    If Trim(buf$) = "Completed: ALTER DATABASE DISMOUNT" Then
        bClosed = True
    End If
    If Trim(buf$) = "Completed: ALTER DATABASE pay4win DISMOUNT" Then
        bClosed = True
    End If

```

```

        DoEvents
    Loop
    Close #fhCheck
    If bClosed = False Then
        ' -----
        ' this means the database was not shutdown properly
        ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
        ' table, then get out.
        ' -----
        rc = WriteLogFile(RES(104))
        rc = WriteNoteOfTheDay(RES(105))
        BailOut (True)
    End If

    fh = FreeFile
    Open App.Path & "\maint\normal.sql" For Output As #fh
    Print #fh, "connect internal/" & g_Password
    Print #fh, "startup pfile=" & App.Path & "\initpcpw.ora"
    Close #fh

    fh = FreeFile
    Open App.Path & "\maint\normal.bat" For Output As #fh
    Print #fh, "set ORACLE_SID=PCPW"
    Print #fh, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\normal.sql"
    Close #fh

    frmStatus.lblTicker.Caption = RES(122)
    frmStatus.Refresh
    DoEvents

    lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")
    ExecDOSCmd (App.Path & "\maint\normal.bat")
    ' -----
    ' Make sure the database is up in normal mode
    ' -----
    fhCheck = FreeFile
    Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
    Seek #fhCheck, lAlertLogLength
    bClosed = False
    Do Until EOF(fhCheck)
        Line Input #fhCheck, buf$
        If Trim(buf$) = "Completed: alter database open" Then
            bClosed = True
        End If
        If Trim(buf$) = "Completed: alter database pay4win open" Then
            bClosed = True
        End If
        DoEvents
    Loop
    Close #fhCheck
    If bClosed = False Then
        ' -----
        ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
        ' table, then get out.
        ' -----
        rc = WriteLogFile(RES(107))
        rc = WriteNoteOfTheDay(RES(108))
        BailOut (True)
    End If

    ' -----
    ' delete all the temporary files, created during the
    ' Health check process
    ' -----
    If g_DEBUG_MODE = False Then
        Call CleanUp
    End If

```

```

frmStatus.lblTicker.Caption = ""
frmStatus.Refresh
DoEvents

rc = CloseLogFile()
Unload frmStatus
End

End Sub

Private Function FixTables() As Boolean

    Dim fh As Integer
    Dim buf As String

    If Dir$(App.Path & "\maint\no_fix.sql", vbNormal) = "" Then
        rc = WriteLogFile(RES(123))
        WriteNoteOfTheDay (RES(110))
        BailOut (True)
    End If

    fh = FreeFile
    Open App.Path & "\maint\no_fix.sql" For Output As #fh
    Print #fh, "/* FIND ALL HIGH RISK TABLES/INDEXES IN MORE THAN 1 EXTENT */"
    Print #fh, ""
    Print #fh, "set termout off"
    Print #fh, "set echo off"
    Print #fh, "set heading off"
    Print #fh, "set pagesize 0"
    Print #fh, "set pause off"
    Print #fh, "set space 0"
    Print #fh, "set verify off"
    Print #fh, "set feed off"
    Print #fh, " "
    Print #fh, ""
    Print #fh, "spool &l\no_fix.out"
    Print #fh, " "
    Print #fh, "column segment_name format a30"
    Print #fh, " "
    Print #fh, "SELECT SEGMENT_NAME, SEGMENT_TYPE, EXTENTS || ' EXTENTS'"
    Print #fh, "FROM DBA_SEGMENTS T2"
    Print #fh, "WHERE OWNER='PCPAYSYS' AND"
    Print #fh, "SEGMENT_TYPE = 'TABLE'"
    Print #fh, "AND EXTENTS > 1"
    Print #fh, "AND SEGMENT_NAME IN"
    Print #fh, "(SELECT SEGMENT_NAME FROM DBA_SEGMENTS T2"
    Print #fh, " Where"
    Print #fh, " SEGMENT_NAME IN"
    Print #fh, " ("

    fhT = FreeFile
    Open App.Path & "\hwbtbl.lst" For Input As #fhT
    Do Until EOF(fhT)
        Line Input #fhT, tBuf$
        Print #fh, " " & tBuf$
    Loop
    Close #fhT

    Print #fh, "    ))"
    Print #fh, "Union"
    Print #fh, "SELECT SEGMENT_NAME, SEGMENT_TYPE, EXTENTS || ' EXTENTS'"
    Print #fh, "FROM DBA_SEGMENTS T2"
    Print #fh, "WHERE OWNER='PCPAYSYS' AND"
    Print #fh, "SEGMENT_TYPE = 'INDEX'"
    Print #fh, "AND EXTENTS > 1"
    Print #fh, "AND SEGMENT_NAME IN"

```

```

Print #fh, "          ("

fhI = FreeFile
Open App.Path & "\hwbidx.lst" For Input As #fhI
Do Until EOF(fhI)
    Line Input #fhI, tBuf$
    Print #fh, "          " & tBuf$
Loop
Close #fhI

Print #fh, "          )"
Print #fh, "/"
Print #fh, ""
Print #fh, "spool off"
Print #fh, "spool &l\nofdone.out"
Print #fh, "select 'nofdone' from dual;"
Print #fh, "spool off"
Print #fh, "exit"
Close #fh

' -----
' first, run the no_fix.sql file to generate
' a list of tables that the Health Check won't
' fix, but should be looked at. This will be
' generated into a no_fix.out file.
' -----
fh = FreeFile
Open App.Path & "\maint\no_fix.bat" For Output As #fh
Print #fh, "set ORACLE_SID=PCPW"
Print #fh, g_szOracleHome & "\sqlplus pcpaysys/" & g_MaintPassword & " @" & App.Path &
"\maint\no_fix.sql " & App.Path & "\maint"
Close #fh

' clear the flag file
rc = RemoveFile(App.Path & "\maint\nofdone.out")
ExecDOSCmd (App.Path & "\maint\no_fix.bat")

frmStatus.lblTicker.Caption = RES(124)
' make sure the prior step is complete before continuing
Do Until Dir$(App.Path & "\maint\nofdone.out", vbNormal) <> ""
    DoEvents
Loop

' -----
' open the no_fix.out file, if there are any
' tables listed, copy the text to the log file
' and generate some warnings....
' -----
If FileLen(App.Path & "\maint\no_fix.out") > 0 Then
    fh = FreeFile
    Open App.Path & "\maint\no_fix.out" For Input As #fh
    Do Until EOF(fh)
        Line Input #fh, buf

        ' strip out multiple spaces
        t1$ = buf
        t2$ = ""
        bSpace = False
        For kk = 1 To Len(t1$)

            If Mid$(t1$, kk, 1) <> " " Or (Mid$(t1$, kk, 1) = " " And bSpace = False)
Then
                t2$ = t2$ & Mid$(t1$, kk, 1)
            End If

            If Mid$(t1$, kk, 1) = " " Then
                bSpace = True
            End If
        Next kk
    Loop
    Close #fh
End If

```

```

        Else
            bSpace = False
        End If

    Next kk

    rc = WriteLogFile(RES(125) & Trim(t2$) & RES(126))
    dWarningCount = dWarningCount + 1
Loop
Close #fh
End If

fh = FreeFile
Open App.Path & "\maint\db_info.bat" For Output As #fh
Print #fh, "set ORACLE_SID=PCPW"
Print #fh, g_szOracleHome & "\sqlplus pcpsys/" & g_MaintPassword & " @" & App.Path &
"\maint\db_info.sql " & App.Path & "\maint"
Close #fh

' clear the flag file
rc = RemoveFile(App.Path & "\maint\infodone.out")
ExecDOSCmd (App.Path & "\maint\db_info.bat")

' make sure the prior step is complete before continuing
frmStatus.lblTicker.Caption = RES(127)
Do Until Dir$(App.Path & "\maint\infodone.out", vbNormal) <> ""
    DoEvents
Loop

fh = FreeFile
Open App.Path & "\maint\bld_anal.sql" For Output As #fh
Print #fh, "set heading off"
Print #fh, "set pagesize 400"
Print #fh, "set pause off"
Print #fh, "set space 0"
Print #fh, "set termout off"
Print #fh, "set feed off"
Print #fh, "spool &l\analyze.sql"
Print #fh, ""
Print #fh, "SELECT 'connect pcpsys/" & g_MaintPassword & "' from dual;"
Print #fh, "select"
Print #fh, "      'ANALYZE TABLE '||table_name||"
Print #fh, "      ' COMPUTE STATISTICS ; '"
Print #fh, "From"
Print #fh, "      user_tables"
Print #fh, "WHERE table_name not in"
Print #fh, "      ("

fhT = FreeFile
Open App.Path & "\hwbtbl.lst" For Input As #fhT
Do Until EOF(fhT)
    Line Input #fhT, tBuf$
    Print #fh, "      " & tBuf$
Loop
Close #fhT

Print #fh, "      )"
Print #fh, ""
Print #fh, "/"
Print #fh, "select 'exit' from dual;"
Print #fh, "spool off"
Print #fh, "spool &l\analdone.out"
Print #fh, "select 'analdone' from dual;"
Print #fh, "spool off"
Print #fh, "exit"
Close #fh

```



```

Print #fh, "          DBA_SEGMENTS T2"
Print #fh, "WHERE T2.OWNER='PCPAYSYS' AND"
Print #fh, "SEGMENT_TYPE = 'TABLE'"
Print #fh, "AND EXTENTS > " & g_dExtents
Print #fh, "AND T1.TABLE_NAME = T2.SEGMENT_NAME"
Print #fh, "AND SEGMENT_NAME NOT IN"
Print #fh, "(SELECT SEGMENT_NAME FROM DBA_SEGMENTS T2"
Print #fh, " Where"
Print #fh, " SEGMENT_NAME IN"
Print #fh, "          ("

fhT = FreeFile
Open App.Path & "\hwbtbl.lst" For Input As #fhT
Do Until EOF(fhT)
    Line Input #fhT, tBuf$
    Print #fh, "          " & tBuf$
Loop
Close #fhT

Print #fh, "          ))"
Print #fh, "ORDER BY SEGMENT_NAME"
Print #fh, "/"
Print #fh, ""
Print #fh, "spool off"
Print #fh, "spool &l\tabdone.out"
Print #fh, "select 'tabdone' from dual;"
Print #fh, "spool off"
Print #fh, ""
Print #fh, "EXIT"
Close #fh

fh = FreeFile
Open App.Path & "\maint\fix_tab.bat" For Output As #fh
Print #fh, "set ORACLE_SID=PCPW"
Print #fh, g_szOracleHome & "\sqlplus pcpaysys/" & g_MaintPassword & " @" & App.Path &
"\maint\fix_tab.sql " & App.Path & "\maint"
Close #fh

' clear the flag file
rc = RemoveFile(App.Path & "\maint\tabdone.out")
ExecDOSCmd (App.Path & "\maint\fix_tab.bat")

' make sure the prior step is complete before continuing
frmStatus.lblTicker.Caption = RES(129)
Do Until Dir$(App.Path & "\maint\tabdone.out", vbNormal) <> ""
    DoEvents
Loop

rc = WriteLogFile(RES(130))
FixTables = True

End Function

Private Function CheckPerf() As Boolean

    Dim fh As Integer
    Dim fhTable As Integer
    Dim buf As String
    Dim recbuf As bufLayout

    Dim di As New clsDiskInfo
    Dim freebytes As Double

    If Dir$(App.Path & "\perf.sql", vbNormal) = "" Then
        rc = WriteLogFile(RES(131))
        WriteNoteOfTheDay (RES(110))
        BailOut (True)
    
```

```

End If

fh = FreeFile
Open App.Path & "\getperf.bat" For Output As #fh
Print #fh, "set ORACLE_SID=PCPW"
Print #fh, g_szOracleHome & "\sqlplus pcpaysys/" & g_MaintPassword & " @" & App.Path &
"\perf.sql " & App.Path & "\"
Close #fh

' clear the flag file
rc = RemoveFile(App.Path & "\perfdone.out")
ExecDOSCmd (App.Path & "\getperf.bat")

' make sure the prior step is complete before continuing
frmStatus.lblTicker.Caption = RES(132)
Do Until Dir$(App.Path & "\perfdone.out", vbNormal) <> ""
    DoEvents
Loop

rc = RemoveFile(App.Path & "\getperf.bat")

On Error GoTo NoPerfTable
fhTable = FreeFile
Open App.Path & "\Perf.tbl" For Input As #fhTable
On Error GoTo 0

fh = FreeFile
On Error GoTo NoPerfLog
Open App.Path & "\perf.out" For Input As #fh
On Error GoTo 0

' -----
' loop through the dynamic array. If there are any
' warnings, simply increment the warning flag
' if any serious errors, set the error flag
' -----
Do Until EOF(fh)

    ' -----
    ' read the next line
    ' -----
    Line Input #fh, buf
    ' -----
    ' build the input buffer by taking the criteria
    ' and the actual amount from the perf.out file (fh)
    ' and the warning and error values from the
    ' perf.tbl file (fhTable)
    ' -----
    recbuf.ItemType = Trim(Mid$(buf, 1, 1))
    recbuf.ItemName = Trim(Mid$(buf, 2, 30))
    recbuf.ActualValue = Trim(Mid$(buf, 32, 7))
    recbuf.CompareType = ""
    recbuf.WarningValue = ""
    recbuf.ErrorValue = ""

    ' rewind the performance criteria lookup table
    Seek #fhTable, 1
    dCriteriaFound = False

    ' throw away first two header lines
    Line Input #fhTable, buf2$
    Line Input #fhTable, buf2$
    Do Until EOF(fhTable)
        Line Input #fhTable, buf2$
        If Trim(Mid$(buf2$, 3, 30)) = Trim(recbuf.ItemName) Then
            recbuf.CompareType = Mid$(buf2$, 34, 1)
            recbuf.WarningValue = Trim(Mid$(buf2$, 36, 5))

```

```

        recbuf.ErrorValue = Trim(Mid$(buf2$, 42, 5))
        dCriteriaFound = True
    End If
Loop

If dCriteriaFound = False Then

    rc = WriteLogFile(RES(133) & recbuf.ItemName & RES(134))
    g_dWarningCount = g_dWarningCount + 1

Else

    ' -----
    ' parse the line, and look for warnings and alerts
    ' -----
    If g_HWBTables = 1 And recbuf.ItemType = "T" Then
        ' tablespaces
        If recbuf.CompareType = "L" Then
            If Val(recbuf.ActualValue) > Val(recbuf.ErrorValue) Then
                rc = WriteLogFile(RES(135) & Trim(recbuf.ItemName) & RES(136))
                g_dErrorCount = g_dErrorCount + 1
            Else
                If Val(recbuf.ActualValue) > Val(recbuf.WarningValue) Then
                    rc = WriteLogFile(RES(137) & Trim(recbuf.ItemName) & RES(138))
                    g_dWarningCount = g_dWarningCount + 1
                End If
            End If
        End If
    Else
        If Val(recbuf.ActualValue) < Val(recbuf.ErrorValue) Then
            rc = WriteLogFile(RES(135) & Trim(recbuf.ItemName) & RES(136))
            g_dErrorCount = g_dErrorCount + 1
        Else
            If Val(recbuf.ActualValue) < Val(recbuf.WarningValue) Then
                rc = WriteLogFile(RES(137) & Trim(recbuf.ItemName) & RES(138))
                m_warnings = m_warnings + 1
                g_dWarningCount = g_dWarningCount + 1
            End If
        End If
    End If
End If
If g_HWBPerf = 1 And recbuf.ItemType = "P" Then
    ' performance criteria
    If recbuf.CompareType = "L" Then
        If Val(recbuf.ActualValue) > Val(recbuf.ErrorValue) Then
            rc = WriteLogFile(RES(139) & Trim(recbuf.ItemName) & RES(140))
            g_dErrorCount = g_dErrorCount + 1
        Else
            If Val(recbuf.ActualValue) > Val(recbuf.WarningValue) Then
                rc = WriteLogFile(RES(141) & Trim(recbuf.ItemName) & RES(142))
                g_dWarningCount = g_dWarningCount + 1
            End If
        End If
    Else
        If Val(recbuf.ActualValue) < Val(recbuf.ErrorValue) Then
            rc = WriteLogFile(RES(139) & Trim(recbuf.ItemName) & RES(140))
            g_dErrorCount = g_dErrorCount + 1
        Else
            If Val(recbuf.ActualValue) < Val(recbuf.WarningValue) Then
                rc = WriteLogFile(RES(141) & Trim(recbuf.ItemName) & RES(142))
                m_warnings = m_warnings + 1
                g_dWarningCount = g_dWarningCount + 1
            End If
        End If
    End If
End If
End If

```

```

Loop

' -----
' close performance log file
' -----
Close #fhTable
Close #fh

' -----
' check physical disk space. if less than 5 MEG free
' alert the user
' -----
For i = 1 To 26

    If di.DriveType(Chr$(64 + i)) = 3 Or di.DriveType(Chr$(64 + i)) = 4 Then
        di.PathName = Chr$(64 + i) + ":\\"
        freebytes = GetDiskFreeSpaceLarge(di.PathName)
        ' -----
        ' adjust freebytes for any dbfs that are already
        ' targetted for this drive
        ' -----
        If freebytes > 0 And freebytes < (g_MEGFree * 1000000) Then
            ' -----
            ' warn the user
            ' -----
            rc = WriteLogFile(RES(143) & di.PathName & RES(144) & Format$(freebytes,
"#,##0") & RES(145))
            g_dWarningCount = g_dWarningCount + 1
        End If
    End If

Next i

On Error GoTo 0
CheckPerf = True

rc = WriteLogFile(RES(146))
Exit Function

NoPerfTable:
On Error GoTo 0
rc = WriteLogFile(RES(147))
g_dErrorCount = g_dErrorCount + 1
Exit Function

NoPerfLog:
Close #fhTable
On Error GoTo 0
rc = WriteLogFile(RES(148))
g_dErrorCount = g_dErrorCount + 1
Exit Function

End Function

Private Function CheckFragAlarms() As Boolean

    Dim fh As Integer
    Dim fh2 As Integer
    Dim buf As String
    Dim tName As String
    Dim tSize As Double
    Dim tTblSpaceName As String
    Dim tTblSpaceSize As Double

    ' -----
    ' open frag alarm log file

```

```

' -----
fh = FreeFile
On Error GoTo NoAlarmLog
Open App.Path & "\maint\fix_tab.out" For Input As #fh
On Error GoTo 0

' -----
' if any entries, FIX 'EM
' -----

If LOF(fh) > 0 Then

    ' -----
    ' for each entry in the alarm log file
    ' -----

    Do Until EOF(fh)

        bExportOk = False
        bDropOk = False
        bImportOk = False

        Line Input #fh, buf

        ' -----
        ' get the table name and the required disk space
        ' for the export file
        ' -----
        tName = Trim(Left$(buf, 30))
        tSize = Val(Trim(Mid$(buf, 31, 16)))
        tTblSpaceName = Trim(Mid$(buf, 47, 30))

        ' -----
        ' make sure there's enough free space in the
        ' tablespace before continuing
        ' -----

        fh2 = FreeFile
        Open App.Path & "\chkfrag.bat" For Output As #fh2
        Print #fh2, "set ORACLE_SID=PCPW"
        Print #fh2, g_szOracleHome & "\sqlplus pcpayssys/" & g_MaintPassword & " @" &
App.Path & "\chkfrag.sql " & tName & " " & App.Path
        Close #fh2

        rc = RemoveFile(App.Path & "\chkfdone.out")
        ExecDOSCmd (App.Path & "\chkfrag.bat")

        ' make sure the prior step is complete before continuing
        frmStatus.lblTicker.Caption = RES(149) & tName & RES(150)
        Do Until Dir$(App.Path & "\chkfdone.out", vbNormal) <> ""
            DoEvents
        Loop

        fh2 = FreeFile
        Open App.Path & "\Chk_sp.out" For Input As #fh2
        Line Input #fh2, tempBuf$
        Close #fh2

        If Val(Trim(tempBuf$)) = 0 Then
            rc = WriteLogFile(RES(151) & tTblSpaceName & RES(152) & tName & RES(153))
            g_dWarningCount = g_dWarningCount + 1
            GoTo NextItem
        End If

        rc = RemoveFile(App.Path & "\chkfrag.bat")
        rc = WriteLogFile(RES(154) & tName & "(" & Format$(tSize, "#,##0") & RES(155))

        frmStatus.lblTicker.Caption = RES(156) & tName
    
```

```

frmStatus.Refresh
DoEvents

' -----
' find a drive that can handle it
' -----
szDrive$ = FindSpace(tSize, "C:")
If szDrive$ = "" Then
    rc = WriteLogFile(RES(157) & Format$(tSize, "#,##0") & RES(158))
    CheckFragAlarms = False
    g_dErrorCount = g_dErrorCount + 1
    Exit Function
Else
    rc = WriteLogFile(RES(159) & tName & RES(160) & szDrive)
End If

' -----
' generate DDL
' -----
' Primary Key
' -----

fh2 = FreeFile
Open App.Path & "\maint\gen_pk.sql" For Output As #fh2
Print #fh2, "set heading off"
Print #fh2, "set pagesize 0"
Print #fh2, "set pause off"
Print #fh2, "set space 0"
Print #fh2, "set termout off"
Print #fh2, "set verify off"
Print #fh2, "set feed off"
Print #fh2, "spool " & App.Path & "\maint\pk.sql"
Print #fh2, "SELECT 'connect pcpayssys/" & g_MaintPassword & "' from dual;"
Print #fh2, "SELECT 'spool &l\" & tName & "_pk.out" & "' from dual;"
Print #fh2, "SELECT 'ALTER TABLE ' || UPPER('" & tName & "') || ' ADD (PRIMARY
KEY (' || column_name"
Print #fh2, "From"
Print #fh2, "    user_cons_columns T1,"
Print #fh2, "    user_constraints T2"
Print #fh2, "Where"
Print #fh2, "    T1.table_name = UPPER('" & tName & "')"
Print #fh2, "    AND constraint_type = 'P'"
Print #fh2, "    AND T1.constraint_name = T2.constraint_name"
Print #fh2, "    AND position = 1"
Print #fh2, "/"
Print #fh2, "SELECT"
Print #fh2, "    ',' || column_name"
Print #fh2, "From"
Print #fh2, "    user_cons_columns T1,"
Print #fh2, "    user_constraints T2"
Print #fh2, "Where"
Print #fh2, "    T1.table_name = UPPER('" & tName & "')"
Print #fh2, "    AND constraint_type = 'P'"
Print #fh2, "    AND T1.constraint_name = T2.constraint_name"
Print #fh2, "    AND position > 1"
Print #fh2, "Order By"
Print #fh2, "    position"
Print #fh2, "/"
Print #fh2, "SELECT '));'"
Print #fh2, "From DUAL"
Print #fh2, "/"
Print #fh2, ""
Print #fh2, "spool off"
Print #fh2, "spool " & App.Path & "\maint\gpkdone.out"
Print #fh2, "select 'gpkdone' from 'ual;"
Print #fh2, "spool off"

Print #fh2, "exit"

```

```

Close #fh2

fh2 = FreeFile
Open App.Path & "\maint\gen_pk.bat" For Output As #fh2
Print #fh2, "set ORACLE_SID=PCPW"
Print #fh2, g_szOracleHome & "\sqlplus pcpaysys/" & g_MaintPassword & " @" &
App.Path & "\maint\gen_pk.sql " & App.Path & "\maint"
Close #fh2

rc = RemoveFile(App.Path & "\maint\gpkdone.out")
ExecDOSCmd (App.Path & "\maint\gen_pk.bat")

' make sure the prior step is complete before continuing
frmStatus.lblTicker.Caption = RES(161) & tName & RES(150)
Do Until Dir$(App.Path & "\maint\gpkdone.out", vbNormal) <> ""
    DoEvents
Loop
' -----
' Make sure the gen_pk.bat process completed w/o errors
' -----
bPk = True
If Dir$(App.Path & "\maint\pk.sql", vbNormal) = "" Then
    bPk = False
Else
    If FileLen(App.Path & "\maint\pk.sql") = 0 Then
        bPk = False
    End If
End If
If bPk = False Then
    rc = WriteLogFile(RES(162) & tName & RES(163))
    g_dErrorCount = g_dErrorCount + 1
    WriteNoteOfTheDay (RES(110))
    GoTo NextItem
End If

rc = WriteLogFile(RES(164) & tName & RES(165))

' -----
' Foreign Key(s)
' -----
fh2 = FreeFile
Open App.Path & "\maint\gen_fk1.sql" For Output As #fh2
Print #fh2, "set heading off"
Print #fh2, "set pagesize 0"
Print #fh2, "set pause off"
Print #fh2, "set space 0"
Print #fh2, "set termout off"
Print #fh2, "set verify off"
Print #fh2, "set feed off"
Print #fh2, "spool " & App.Path & "\maint\fk1.sql"
Print #fh2, "SELECT 'spool " & App.Path & "\maint\fk.sql' from dual;"

Print #fh2, "SELECT 'set heading off' from dual;"
Print #fh2, "SELECT 'set pagesize 0' from dual;"
Print #fh2, "SELECT 'set pause off' from dual;"
Print #fh2, "SELECT 'set space 0' from dual;"
Print #fh2, "SELECT 'set termout off' from dual;"
Print #fh2, "SELECT 'set verify off' from dual;"
Print #fh2, "SELECT 'set feed off' from dual;"

Print #fh2, "SELECT 'select ''connect pcpaysys/" & g_MaintPassword & "' from
dual;' from dual;"
Print #fh2, "SELECT 'select ''spool " & App.Path & "\maint\" & tName &
"_fk.log" & "' from dual;' from dual;"
Print #fh2, ""
Print #fh2, "/* Generate all Parent Foreign Keys */"
Print #fh2, ""

```

```

        Print #fh2, "SELECT '@' & App.Path & "\maint\Gen_fk2.sql ' || UPPER(' & tName
& "') || ' ' || constraint_name" & " || ' ' || ' & App.Path & "\maint"
        Print #fh2, "From"
        Print #fh2, "user_constraints"
        Print #fh2, "Where"
        Print #fh2, "table_name = UPPER(' & tName & ')"
        Print #fh2, "AND constraint_type = 'R'"
        Print #fh2, "ORDER BY constraint_name"
        Print #fh2, "/"
        Print #fh2, ""
        Print #fh2, "/* Generate all Children Foreign Keys */"
        Print #fh2, ""
        Print #fh2, "SELECT '@' & App.Path & "\maint\Gen_fk2.sql ' || t1.table_name ||
' ' || t1.constraint_name" & " || ' ' || ' & App.Path & "\maint"
        Print #fh2, "From"
        Print #fh2, "user_constraints t1,"
        Print #fh2, "user_constraints t2"
        Print #fh2, "Where"
        Print #fh2, "t2.table_name = ' & tName & ' and t2.constraint_type = 'P'"
        Print #fh2, "AND t2.constraint_name = t1.r_constraint_name"
        Print #fh2, "ORDER BY t1.table_name"
        Print #fh2, "/"
        Print #fh2, "SELECT 'select 'spool off' FROM dual;' from dual;"
        Print #fh2, ""
        Print #fh2, "SELECT 'spool off' FROM dual;"
        Print #fh2, "SELECT 'select 'spool off' FROM dual;' from dual;"
        Print #fh2, "SELECT 'spool ' & App.Path & "\maint\fk2done.out' from dual;"
        Print #fh2, "SELECT 'select 'fk2done' from dual;' from dual;"
        Print #fh2, "SELECT 'spool off' FROM dual;"
        Print #fh2, "SELECT 'exit' FROM dual;"
        Print #fh2, ""
        Print #fh2, "spool off"
        Print #fh2, "spool " & App.Path & "\maint\fk1done.out"
        Print #fh2, "select 'fk1done' from dual;"
        Print #fh2, "spool off"
        Print #fh2, "exit"
        Close #fh2

        fh2 = FreeFile
        Open App.Path & "\maint\gen_fk1.bat" For Output As #fh2
        Print #fh2, "set ORACLE_SID=PCPW"
        Print #fh2, g_szOracleHome & "\sqlplus pcpsys/" & g_MaintPassword & " @" &
App.Path & "\maint\gen_fk1.sql"
        Close #fh2

        ' clear the flag file
        rc = RemoveFile(App.Path & "\maint\fk1done.out")
        ExecDOSCmd (App.Path & "\maint\gen_fk1.bat")

        rc = WriteLogFile(RES(166) & tName & RES(167))

        fh2 = FreeFile
        Open App.Path & "\maint\gen_fk.bat" For Output As #fh2
        Print #fh2, "set ORACLE_SID=PCPW"
        Print #fh2, g_szOracleHome & "\sqlplus pcpsys/" & g_MaintPassword & " @" &
App.Path & "\maint\fk1.sql"
        Close #fh2

        ' make sure the prior step is complete before continuing
        frmStatus.lblTicker.Caption = RES(168) & tName & RES(169)
        Do Until Dir$(App.Path & "\maint\fk1done.out", vbNormal) <> ""
            DoEvents
        Loop
        ' -----
        ' Make sure the gen_fk1.bat process completed w/o errors
        ' -----
        bFK = True

```

```

If Dir$(App.Path & "\maint\fk1.sql", vbNormal) = "" Then
    bFK = False
Else
    If FileLen(App.Path & "\maint\fk1.sql") = 0 Then
        bFK = False
    End If
End If
If bFK = False Then
    rc = WriteLogFile(RES(170) & tName & RES(171))
    g_dErrorCount = g_dErrorCount + 1
    WriteNoteOfTheDay (RES(110))
    GoTo NextItem
End If

rc = RemoveFile(App.Path & "\maint\fk2done.out")
ExecDOSCmd (App.Path & "\maint\gen_fk.bat")

' make sure the prior step is complete before continuing
frmStatus.lblTicker.Caption = RES(168) & tName & RES(172)
Do Until Dir$(App.Path & "\maint\fk2done.out", vbNormal) <> ""
    DoEvents
Loop
' -----
' Make sure the gen_fk.bat process completed w/o errors
' -----
bFK = True
If Dir$(App.Path & "\maint\fk.sql", vbNormal) = "" Then
    bFK = False
Else
    If FileLen(App.Path & "\maint\fk.sql") = 0 Then
        bFK = False
    End If
End If
If bFK = False Then
    rc = WriteLogFile(RES(173) & tName)
    g_dWarningCount = g_dWarningCount + 1
End If

rc = WriteLogFile(RES(174) & tName & RES(175))

' -----
' do the export
' -----
' create exp.sql in the maint folder and execute it using
' SVRMGR23 (NT), the export statement looks like...
fh2 = FreeFile
Open App.Path & "\maint\export.bat" For Output As #fh2
Print #fh2, "set ORACLE_SID=PCPW"
Print #fh2, g_szOracleHome & "\exp73 pcphys/" & g_MaintPassword & "
constraints=n tables=(" & tName & ") file=" & szDrive$ & "\export\" & tName & ".dmp log="
& g_szMaint & "\" & tName & "_exp.log"
Close #fh2

' now, execute the bat file just created in the step above
bDir = False
If Dir$(szDrive$ & "\export", vbDirectory) = "" Then
    MkDir szDrive$ & "\export"
    bDir = True
End If
ExecDOSCmd (App.Path & "\maint\export.bat")

' -----
' Make sure the export.bat process completed w/o errors
' -----
fhLog = FreeFile
On Error GoTo NoExportLog
Open g_szMaint & "\" & tName & "_exp.log" For Input As #fhLog

```

```

On Error GoTo 0
bOk = False
Do Until EOF(fhLog)
    Line Input #fhLog, buf$
    If Trim(buf$) = "Export terminated successfully without warnings." Then
        bOk = True
    End If
Loop
Close #fhLog
If Not bOk Then
    rc = WriteLogFile(RES(176) & g_szMaint & "\" & tName & RES(177))
    g_dErrorCount = g_dErrorCount + 1
    WriteNoteOfTheDay (RES(110))
    GoTo NextItem
End If

bExportOk = True

rc = WriteLogFile(RES(178))

GoTo DropTheTable

NoExportLog:
On Error GoTo 0
rc = WriteLogFile(RES(179))
g_dErrorCount = g_dErrorCount + 1
WriteNoteOfTheDay (RES(110))
GoTo NextItem

DropTheTable:
' -----
' drop the table
' -----

fh2 = FreeFile
Open App.Path & "\maint\drop.sql" For Output As #fh2
Print #fh2, "connect pcpayssys/" & g_MaintPassword
Print #fh2, "spool '" & App.Path & "\maint\drop.log'"
Print #fh2, "DROP TABLE " & tName & " CASCADE CONSTRAINTS;"
Print #fh2, "ALTER TABLESPACE " & tTblSpaceName & " COALESCE;"
Close #fh2

fh2 = FreeFile
Open App.Path & "\maint\drop.bat" For Output As #fh2
Print #fh2, "set ORACLE_SID=PCPW"
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\drop.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\drop.bat")
' -----
' Make sure the drop.bat process completed w/o errors
' -----

fhCheck = FreeFile
sCount = 0
Open App.Path & "\maint\drop.log" For Input As #fhCheck
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$
    If Trim(buf$) = "Statement processed." Then
        sCount = sCount + 1
    End If
    DoEvents
Loop
Close #fhCheck
If sCount <> 2 Then
    rc = WriteLogFile(RES(180) & tName)
    g_dErrorCount = g_dErrorCount + 1
    WriteNoteOfTheDay (RES(110))
    GoTo NextItem

```

```

End If

bDropOk = True

rc = WriteLogFile(RES(181) & tName & RES(182))

' -----
' import the exported data
' -----
fh2 = FreeFile
Open App.Path & "\maint\import.bat" For Output As #fh2
Print #fh2, "set ORACLE_SID=PCPW"
Print #fh2, g_szOracleHome & "\imp73 pcpayssys/" & g_MaintPassword & "
tables=(" & tName & ") file=" & szDrive$ & "\export\" & tName & ".dmp log=" & g_szMaint &
"\\" & tName & "_imp.log"
Close #fh2

ExecDOSCmd (App.Path & "\maint\import.bat")
' -----
' Make sure the import.bat process completed w/o errors
' -----
fhCheck = FreeFile
Open App.Path & "\maint\" & tName & "_imp.log" For Input As #fhCheck
bOk = False
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$
    If Trim(buf$) = "Import terminated successfully without warnings."
Then
        bOk = True
    End If
    DoEvents
Loop
Close #fhCheck
If bOk = False Then
    rc = WriteLogFile(RES(183) & tName)
    g_dErrorCount = g_dErrorCount + 1
    WriteNoteOfTheDay (RES(110))
    BailOut (False)
End If

bImportOk = True

rc = WriteLogFile(RES(184))

frmStatus.Picture1(2).Visible = True
frmStatus.Label2(2).FontBold = False
frmStatus.Label2(3).FontBold = True
frmStatus.Refresh

' -----
' use generated DDL to recreate constraints
' -----
fh2 = FreeFile
Open App.Path & "\maint\ddlpk.bat" For Output As #fh2
Print #fh2, "set ORACLE_SID=PCPW"
Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\pk.sql"
Close #fh2

ExecDOSCmd (App.Path & "\maint\ddlpk.bat")
' -----
' Make sure the ddlpk.bat process completed w/o errors
' -----
fhCheck = FreeFile
Open App.Path & "\maint\" & tName & "_pk.out" For Input As #fhCheck
bOk = False
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$

```

```

        If UCase$(Trim(buf$)) = "STATEMENT PROCESSED." Then
            bOk = True
        End If
        DoEvents
    Loop
    Close #fhCheck
    If bOk = False Then
        rc = WriteLogFile(RES(185) & tName & RES(186))
        g_dErrorCount = g_dErrorCount + 1
        WriteNoteOfTheDay (RES(110))
        BailOut (False)
    End If

    rc = WriteLogFile(RES(187))

    fh2 = FreeFile
    Open App.Path & "\maint\ddlfk.bat" For Output As #fh2
    Print #fh2, "set ORACLE_SID=PCPW"
    Print #fh2, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\fk.sql"
    Close #fh2

    ExecDOSCmd (App.Path & "\maint\ddlfk.bat")
    ' -----
    ' Make sure the ddlfk.bat process completed w/o errors
    ' there should be one 'Table altered' for each foreign key
    ' -----
    On Error GoTo NoForeignKeys
    fhCheck = FreeFile
    bOk = True
    Open App.Path & "\maint\" & tName & "_fk.log" For Input As #fhCheck
    On Error Resume Next
    Do Until EOF(fhCheck)
        Line Input #fhCheck, buf$
        If Mid(buf$, 1, 4) = "ORA-" Then
            bOk = False
        End If
        DoEvents
    Loop
    Close #fhCheck
    On Error GoTo 0
    If bOk = False Then
        rc = WriteLogFile(RES(185) & tName & RES(188))
        g_dErrorCount = g_dErrorCount + 1
        WriteNoteOfTheDay (RES(110))
        BailOut (False)
    End If

    rc = WriteLogFile(RES(189))
    GoTo NextItem

```

NoForeignKeys:

```

    On Error GoTo 0
    rc = WriteLogFile(RES(190) & tName)

```

NextItem:

```

    On Error GoTo 0
    ' -----
    ' cleanup and get ready for the next table
    ' -----
    If Dir$(szDrive$ & "\export\" & tName & ".dmp", vbNormal) <> "" Then
        If bImportOk = True Then
            rc = RemoveFile(szDrive$ & "\export\" & tName & ".dmp")
            'If bDir Then
            '    Rmdir szDrive$ & "\export"
            'End If
        End If
    End If

```

```

        End If

    Loop

    frmStatus.lblTicker.Caption = ""

    rc = WriteLogFile(RES(191))

Else

    rc = WriteLogFile(RES(192))

End If

frmStatus.Picture1(3).Visible = True
frmStatus.Label2(3).FontBold = False
frmStatus.Label2(4).FontBold = True
frmStatus.Refresh

' -----
' close frag alarm log file
' -----

Close #fh
On Error GoTo 0

' -----
' now, check to see if any indexes need to be
' rebuilt...
' -----

fh2 = FreeFile
Open App.Path & "\maint\fix_idx.sql" For Output As #fh2
Print #fh2, "/* FIND ALL TABLES IN MORE THAN " & g_dExtents & " EXTENT WHICH SHOULD BE
REORGANIZED */"
Print #fh2, "set termout off"
Print #fh2, "set echo off"
Print #fh2, "set heading off"
Print #fh2, "set pagesize 0"
Print #fh2, "set pause off"
Print #fh2, "set space 0"
Print #fh2, "set verify off"
Print #fh2, "set feed off"
Print #fh2, " "
Print #fh2, "spool " & App.Path & "\maint\Rbld_idx.sql"
Print #fh2, " "
Print #fh2, "SELECT 'spool " & App.Path & "\maint\rbld_idx.log' from dual;"
Print #fh2, "SELECT 'ALTER TABLESPACE INDEX_DATA1 COALESCE;' FROM DUAL"
Print #fh2, "/"
Print #fh2, "SELECT 'ALTER TABLESPACE INDEX_DATA2 COALESCE;' FROM DUAL"
Print #fh2, "/"
Print #fh2, "SELECT 'ALTER TABLESPACE INDEX_DATA3 COALESCE;' FROM DUAL"
Print #fh2, "/"
Print #fh2, " "
Print #fh2, " "
Print #fh2, "SELECT 'ALTER INDEX ' || SEGMENT_NAME || ' REBUILD '"
Print #fh2, "|| 'STORAGE(INITIAL ' || T1.BYTES || ' NEXT ' ||
(2048*FLOOR(T1.BYTES/8192)))"
Print #fh2, "|| ' ) TABLESPACE ' || T1.TABLESPACE_NAME || ';' "
Print #fh2, "FROM DBA_SEGMENTS T1"
Print #fh2, "Where"
Print #fh2, "SEGMENT_TYPE = 'INDEX'"
Print #fh2, "AND OWNER = 'PCPAYSYS'"
Print #fh2, "AND EXTENTS > " & g_dExtents
Print #fh2, "AND SEGMENT_NAME NOT IN"
Print #fh2, " ("

fhI = FreeFile

```

```

Open App.Path & "\hwbidx.lst" For Input As #fhI
Do Until EOF(fhI)
    Line Input #fhI, tBuf$
    Print #fh2, "          " & tBuf$
Loop
Close #fhI

Print #fh2, "          )"
Print #fh2, "AND T1.BYTES <"
Print #fh2, "(SELECT SUM(BYTES)"
Print #fh2, "FROM DBA_FREE_SPACE T2"
Print #fh2, "WHERE T1.TABLESPACE_NAME = T2.TABLESPACE_NAME)"
Print #fh2, "ORDER BY SEGMENT_NAME"
Print #fh2, "/"
Print #fh2, "SELECT 'spool off' FROM dual;"
Print #fh2, "SELECT 'spool " & App.Path & "\maint\rbldone.out' from dual;"
Print #fh2, "SELECT 'select 'rbldone'' FROM dual;' from dual;"
Print #fh2, "SELECT 'spool off' FROM dual;"

Print #fh2, "SELECT 'exit' FROM dual;"
Print #fh2, "          "
Print #fh2, "spool off"
Print #fh2, "          "
Print #fh2, "spool " & App.Path & "\maint\Alrt_rbld.out"
Print #fh2, "          "
Print #fh2, "SELECT SEGMENT_NAME || ' ** Can not rebuild - Not enough Space ***"
Print #fh2, "FROM DBA_SEGMENTS T1"
Print #fh2, "Where"
Print #fh2, "SEGMENT_TYPE = 'INDEX'"
Print #fh2, "AND OWNER = 'PCPAYSYS'"
Print #fh2, "AND EXTENTS > " & g_dExtents
Print #fh2, "AND SEGMENT_NAME NOT IN"
Print #fh2, "          ("

fhI = FreeFile
Open App.Path & "\hwbidx.lst" For Input As #fhI
Do Until EOF(fhI)
    Line Input #fhI, tBuf$
    Print #fh2, "          " & tBuf$
Loop
Close #fhI

Print #fh2, "          )"
Print #fh2, "AND T1.BYTES >"
Print #fh2, "(SELECT SUM(BYTES)"
Print #fh2, "FROM DBA_FREE_SPACE T2"
Print #fh2, "WHERE T1.TABLESPACE_NAME = T2.TABLESPACE_NAME)"
Print #fh2, "ORDER BY SEGMENT_NAME"
Print #fh2, "/"
Print #fh2, "          "

Print #fh2, "spool off"
Print #fh2, "spool " & App.Path & "\maint\idxdone.out"
Print #fh2, "select 'idxdone' from dual;"
Print #fh2, "spool off"
Print #fh2, "SELECT 'exit' FROM dual;"
Print #fh2, "          "
Print #fh2, "EXIT"
Close #fh2

fh2 = FreeFile
Open App.Path & "\maint\fix_idx.bat" For Output As #fh2
Print #fh2, "set ORACLE_SID=PCPW"
Print #fh2, g_szOracleHome & "\sqlplus pcpay\ys/" & g_MaintPassword & " @" & App.Path
& "\maint\fix_idx.sql"
Close #fh2

```

```

rc = RemoveFile(App.Path & "\maint\idxdone.out")
ExecDOSCmd (App.Path & "\maint\fix_idx.bat")

' make sure the prior step is complete before continuing
frmStatus.lblTicker.Caption = RES(193)
Do Until Dir$(App.Path & "\maint\idxdone.out", vbNormal) <> ""
    DoEvents
Loop
' -----
' Make sure the fix_idx.bat process completed w/o errors
' -----
If Dir$(App.Path & "\maint\rbld_idx.sql", vbNormal) = "" Then
    rc = WriteLogFile(RES(194))
    g_dWarningCount = g_dWarningCount + 1
    WriteNoteOfTheDay (RES(110))
Else
    ' before we run the rbld_idx.sql script, open it and strip out the index names
    fhIdxName = FreeFile
    Open App.Path & "\maint\rbld_idx.sql" For Input As #fhIdxName
    Line Input #fhIdxName, fhIdxName_buf$
    Do Until EOF(fhIdxName)
        If Mid$(fhIdxName_buf$, 1, 12) = "ALTER INDEX " Then
            rc = WriteLogFile(RES(195) & Mid$(fhIdxName_buf$, 13))
        End If
        Line Input #fhIdxName, fhIdxName_buf$
    Loop
    Close #fhIdxName

    fh2 = FreeFile
    Open App.Path & "\maint\Rbld_idx.bat" For Output As #fh2
    Print #fh2, "set ORACLE_SID=PCPW"
    Print #fh2, g_szOracleHome & "\sqlplus pcpayssys/" & g_MaintPassword & " @" &
App.Path & "\maint\Rbld_idx.sql"
    Close #fh2

    rc = RemoveFile(App.Path & "\maint\rbldone.out")
    ExecDOSCmd (App.Path & "\maint\Rbld_idx.bat")

    ' make sure the prior step is complete before continuing
    frmStatus.lblTicker.Caption = RES(196)
    Do Until Dir$(App.Path & "\maint\rbldone.out", vbNormal) <> ""
        DoEvents
    Loop
    ' -----
    ' TODO: Make sure the rbld_idx.bat process completed w/o errors
    ' -----
    fhCheck = FreeFile
    Open App.Path & "\maint\rbld_idx.log" For Input As #fhCheck
    bOk = True
    Do Until EOF(fhCheck)
        Line Input #fhCheck, buf$
        If Mid(buf$, 1, 4) = "ORA-" Then
            bOk = False
        End If
        DoEvents
    Loop
    Close #fhCheck
    If bOk = False Then
        rc = WriteLogFile(RES(197))
        g_dErrorCount = g_dErrorCount + 1
        WriteNoteOfTheDay (RES(110))
        BailOut (False)
    End If

    rc = WriteLogFile(RES(198))
End If

```

```

    frmStatus.Picture1(4).Visible = True
    frmStatus.Label2(4).FontBold = False
    frmStatus.Refresh

    CheckFragAlarms = True
    Exit Function

NoAlarmLog:
    On Error GoTo 0
    rc = WriteLogFile(RES(199))
    g_dErrorCount = g_dErrorCount + 1
    CheckFragAlarms = False
    Exit Function

End Function

Function FindSpace(spaceNeeded As Double, startingDrive As String) As String
    Dim di As New clsDiskInfo
    Dim freebytes As Double

    ' -----
    ' see if fn can fit on fdr ( size is ns )
    ' -----
    freebytes = GetDiskFreeSpaceLarge(startingDrive)
    If freebytes > spaceNeeded Then
        ' -----
        ' it fits, so just put it here
        ' -----
        FindSpace = startingDrive
        Exit Function
    End If

    ' -----
    ' doesn't fit, so check other drives
    ' -----
    dbFound = False
    For i = 1 To 26

        If di.DriveType(Chr$(64 + i)) = 3 Or di.DriveType(Chr$(64 + i)) = 4 Then
            di.PathName = Chr$(64 + i) + ":\\"
            freebytes = GetDiskFreeSpaceLarge(di.PathName)
            ' -----
            ' adjust freebytes for any dbfs that are already
            ' targetted for this drive
            ' -----
            If freebytes > spaceNeeded Then
                ' -----
                ' it fits here, so put it here
                ' -----
                dbFound = True
                Exit Function
            End If
        End If
    Next i

    If dbFound = False Then
        FindSpace = ""
    Else
        FindSpace = di.PathName
    End If

End Function

```

```
Public Function GetDiskFreeSpaceLarge(DriveLetter As String) As Double
```

```
    Dim hdb As Integer
    Dim bf As Double
    Dim buf As String
    Dim buf2 As String
    Dim fh As Integer
```

```
    fh = FreeFile
    On Error GoTo CantWrite
    Open DriveLetter + "test.txt" For Output As #fh
    On Error GoTo 0
    Print #fh, "Testing"
    Close #fh
```

```
    ' -----
    ' execute both command.com and cmd.com. If running on Win95
    ' the command.com will work, and cmd.com will fail. On WinNT4.0
    ' both will work, but the correct output of cmd.com will overwrite
    ' the incorrect output of command.com. This way the end result
    ' will be correct regardless of OS...
    ' -----
```

```
    ExecDOSCmd ("command.com /c dir " + DriveLetter + "test.txt > c:\dbsizer.lst")
    ExecDOSCmd ("cmd /c dir " + DriveLetter + "test.txt > c:\dbsizer.lst")
    hdb = FreeFile
    Open "c:\dbsizer.lst" For Input As #hdb
    Do Until EOF(hdb)
        Line Input #hdb, buf
        idx = InStr(UCase$(buf), UCase$(RES(210)))
        If idx > 0 Then
            buf = Left$(buf, idx - 2)
            For i% = Len(buf) To 1 Step -1
                If Mid$(buf, i%, 1) = " " Then
                    buf = Mid$(buf, i% + 1)
                    Exit For
                End If
            Next i%
        End If
    Loop
    Close #hdb
    rc = RemoveFile("c:\dbsizer.lst")
    rc = RemoveFile(DriveLetter & "test.txt")
```

```
    buf2 = ""
    For i% = 1 To Len(buf)
        thischar = Mid$(buf, i%, 1)
        If thischar <> " " And thischar <> RES(211) Then
            buf2 = buf2 + thischar
        End If
    Next i%
```

```
    GetDiskFreeSpaceLarge = Val(buf2)
    Exit Function
```

```
CantWrite:
    On Error GoTo 0
    GetDiskFreeSpaceLarge = 0
```

```
End Function
```

```
Public Function WriteNoteOfTheDay(szMsg As String) As Boolean
```

```
    Dim fh As Integer
```

```
    If UCase$(g_WriteNoteoftheDay) = "FALSE" Then
```

```

        WriteNoteOfTheDay = True
        Exit Function
    End If

    If szMsg = "" Then
        szMsg = RES(110)
    End If

    fh = FreeFile
    Open App.Path & "\maint\notd.sql" For Output As #fh
    Print #fh, "connect pcpayssys/" & g_MaintPassword & ";"
    Print #fh, "execute p_modify_postnote('" & szMsg & "','ADD');"
    Print #fh, "exit;"
    Close #fh

    fh = FreeFile
    Open App.Path & "\maint\notd.bat" For Output As #fh
    Print #fh, "set ORACLE_SID=PCPW"
    Print #fh, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\notd.sql"
    Close #fh

    ExecDOSCmd (App.Path & "\maint\notd.bat")
    WriteNoteOfTheDay = True

End Function

Public Function ClearNoteOfTheDay(szMsg As String) As Boolean

    Dim fh As Integer

    If szMsg = "" Then
        szMsg = RES(110)
    End If

    fh = FreeFile
    Open App.Path & "\maint\notd.sql" For Output As #fh
    Print #fh, "connect pcpayssys/" & g_MaintPassword & ";"
    Print #fh, "execute p_modify_postnote('" & szMsg & "','DEL');"
    Print #fh, "exit;"
    Close #fh

    fh = FreeFile
    Open App.Path & "\maint\notd.bat" For Output As #fh
    Print #fh, "set ORACLE_SID=PCPW"
    Print #fh, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\notd.sql"
    Close #fh

    ExecDOSCmd (App.Path & "\maint\notd.bat")
    ClearNoteOfTheDay = True

End Function

Public Function RemoveFile(szFile As String) As Boolean

    frmStatus.lblTicker.Caption = "Removing " & szFile
    frmStatus.Refresh
    DoEvents

    On Error GoTo CannotRemoveFile
    If Dir$(szFile, vbNormal) <> "" Then
        Kill szFile
    End If
    On Error GoTo 0
    RemoveFile = True
    frmStatus.lblTicker.Caption = ""
    frmStatus.Refresh
    DoEvents

```

```

Exit Function

CannotRemoveFile:
    On Error GoTo 0
    RemoveFile = False
    frmStatus.lblTicker.Caption = ""
    frmStatus.Refresh
    DoEvents
    Exit Function

End Function

Public Sub BailOut(ReStart As Boolean)

    ' -----
    ' all entries processed, so shutdown the database
    ' and bring it back up in normal mode.
    ' -----
    frmStatus.lblTicker.Caption = RES(103)
    frmStatus.Refresh
    DoEvents

    lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")
    ExecDOSCmd (App.Path & "\maint\shutdown.bat")
    ' -----
    ' Make sure the database is shutdown
    ' -----

    fhCheck = FreeFile
    frmStatus.lblTicker.Caption = ""
    Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
    Seek #fhCheck, lAlertLogLength
    bClosed = False
    Do Until EOF(fhCheck)
        Line Input #fhCheck, buf$
        If Trim(buf$) = "Completed: ALTER DATABASE DISMOUNT" Then
            bClosed = True
        End If
        If Trim(buf$) = "Completed: ALTER DATABASE pay4win DISMOUNT" Then
            bClosed = True
        End If
        DoEvents
    Loop
    Close #fhCheck
    If bClosed = False Then
        ' -----
        ' this means the database was not shutdown properly
        ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
        ' table, then get out.
        ' -----
        rc = WriteLogFile(RES(104))
        rc = WriteNoteOfTheDay(RES(105))
    End If

    fh = FreeFile
    Open App.Path & "\maint\normal.sql" For Output As #fh
    Print #fh, "connect internal/" & g_Password
    Print #fh, "startup pfile=" & App.Path & "\initpcpw.ora"
    Close #fh

    If ReStart Then
        fh = FreeFile
        Open App.Path & "\maint\normal.bat" For Output As #fh
        Print #fh, "set ORACLE_SID=PCPW"
        Print #fh, g_szOracleHome & "\svrmgr23 @" & App.Path & "\maint\normal.sql"
        Close #fh
    End If

```

```

frmStatus.lblTicker.Caption = RES(200)
frmStatus.Refresh
DoEvents

lAlertLogLength = FileLen(App.Path & "..\log\pcpwALRT.log")
ExecDOSCmd (App.Path & "\maint\normal.bat")
' -----
' Make sure the database is up in normal mode
' -----
fhCheck = FreeFile
frmStatus.lblTicker.Caption = ""
Open App.Path & "..\log\pcpwALRT.log" For Input As #fhCheck
Seek #fhCheck, lAlertLogLength
bClosed = False
Do Until EOF(fhCheck)
    Line Input #fhCheck, buf$
    If Trim(buf$) = "Completed: alter database open" Then
        bClosed = True
    End If
    If Trim(buf$) = "Completed: alter database pay4win open" Then
        bClosed = True
    End If
    DoEvents
Loop
Close #fhCheck
If bClosed = False Then
    ' -----
    ' note it in the HWB.LOG and the NOTE_OF_THE_DAY
    ' table, then get out.
    ' -----
    rc = WriteLogFile(RES(107))
    rc = WriteNoteOfTheDay(RES(108))
End If
End If

' -----
' delete all the temporary files, created during the
' Health check process
' -----
If g_DEBUG_MODE = False Then
    Call CleanUp
End If

frmStatus.lblTicker.Caption = ""
frmStatus.Refresh
DoEvents

rc = CloseLogFile()
Unload frmStatus
End

End Sub

Public Sub CleanUp()

    rc = RemoveFile(App.Path & "\maint\perf.b...")
    rc = RemoveFile(App.Path & "\maint\gen_pk.sql")
    rc = RemoveFile(App.Path & "\maint\gen_pk.bat")
    rc = RemoveFile(App.Path & "\maint\gen_fk1.sql")
    rc = RemoveFile(App.Path & "\maint\gen_fk1.bat")
    rc = RemoveFile(App.Path & "\maint\gen_fk.bat")
    rc = RemoveFile(App.Path & "\maint\fk1.sql")
    rc = RemoveFile(App.Path & "\maint\fk.sql")
    rc = RemoveFile(App.Path & "\maint\fix_idx.sql")
    rc = RemoveFile(App.Path & "\maint\fix_idx.bat")
    rc = RemoveFile(App.Path & "\maint\fix_tab.bat")

```

```

rc = RemoveFile(App.Path & "\maint\fix_tab.sql")
rc = RemoveFile(App.Path & "\maint\pk.sql")
rc = RemoveFile(App.Path & "\maint\export.bat")
rc = RemoveFile(App.Path & "\maint\Drop.sql")
rc = RemoveFile(App.Path & "\maint\Drop.bat")
rc = RemoveFile(App.Path & "\maint\Drop.Log")
rc = RemoveFile(App.Path & "\maint\ddlpk.bat")
rc = RemoveFile(App.Path & "\maint\ddlfk.bat")
rc = RemoveFile(App.Path & "\maint\notd.sql")
rc = RemoveFile(App.Path & "\maint\notd.bat")
rc = RemoveFile(App.Path & "\maint\import.bat")
rc = RemoveFile(App.Path & "\maint\Rbld_idx.bat")
rc = RemoveFile(App.Path & "\maint\Rbld_idx.sql")
rc = RemoveFile(App.Path & "\maint\shutdown.sql")
rc = RemoveFile(App.Path & "\maint\shutdown.bat")
rc = RemoveFile(App.Path & "\maint\normal.bat")
rc = RemoveFile(App.Path & "\maint\normal.sql")
rc = RemoveFile(App.Path & "\maint\restrict.bat")
rc = RemoveFile(App.Path & "\maint\restrict.sql")
rc = RemoveFile(App.Path & "\maint\analyze.bat")
rc = RemoveFile(App.Path & "\maint\analyze.sql")
rc = RemoveFile(App.Path & "\maint\bld_anal.sql")
rc = RemoveFile(App.Path & "\maint\bld_anal.bat")
rc = RemoveFile(App.Path & "\maint\db_info.bat")
rc = RemoveFile(App.Path & "\maint\no_fix.bat")

On Error GoTo NoMaintFilesToDelete
Kill App.Path & "\maint\*.out"

NoMaintFilesToDelete:

    On Error GoTo NoAdminFilesToDelete
    Kill App.Path & "\*.out"

NoAdminFilesToDelete:

    On Error GoTo 0

End Sub

Function StrEncode(s As String, key As Long) As String

'Written by Gary Ardell.
'free from all copyright restrictions

Dim N As Long, i As Long, ss As String
Dim k1 As Long, k2 As Long, k3 As Long, k4 As Long, t As Long
Dim salt As Boolean
Static saltvalue As String * 4

salt = False

If salt Then
    For i = 1 To 4
        t = 100 * (1 + Asc(Mid(saltvalue, i, 1))) * Rnd() * (Timer + 1)
        Mid(saltvalue, i, 1) = Chr(t Mod 256)
    Next
    s = Mid(saltvalue, 1, 2) & s & Mid(saltvalue, 3, 2)
End If

N = Len(s)
ss = Space(N)
ReDim sn(N) As Long

k1 = 11 + (key Mod 233): k2 = 7 + (key Mod 239)
k3 = 5 + (key Mod 241): k4 = 3 + (key Mod 251)

```

```

For i = 1 To N: sn(i) = Asc(Mid(s, i, 1)): Next i

For i = 2 To N: sn(i) = sn(i) Xor sn(i - 1) Xor ((k1 * sn(i - 1)) Mod 256): Next
For i = N - 1 To 1 Step -1: sn(i) = sn(i) Xor sn(i + 1) Xor (k2 * sn(i + 1)) Mod 256: Next
For i = 3 To N: sn(i) = sn(i) Xor sn(i - 2) Xor (k3 * sn(i - 1)) Mod 256: Next
For i = N - 2 To 1 Step -1: sn(i) = sn(i) Xor sn(i + 2) Xor (k4 * sn(i + 1)) Mod 256: Next

For i = 1 To N: Mid(ss, i, 1) = Chr(sn(i)): Next i

StrEncode = ss
saltvalue = Mid(ss, Len(ss) / 2, 4)

End Function

Function StrDecode(s As String, key As Long) As String

'Written by Gary Ardell.
'free from all copyright restrictions

Dim N As Long, i As Long, ss As String
Dim k1 As Long, k2 As Long, k3 As Long, k4 As Long
Dim salt As Boolean

salt = False

N = Len(s)
ss = Space(N)
ReDim sn(N) As Long

k1 = 11 + (key Mod 233): k2 = 7 + (key Mod 239)
k3 = 5 + (key Mod 241): k4 = 3 + (key Mod 251)

For i = 1 To N: sn(i) = Asc(Mid(s, i, 1)): Next

For i = 1 To N - 2: sn(i) = sn(i) Xor sn(i + 2) Xor (k4 * sn(i + 1)) Mod 256: Next
For i = N To 3 Step -1: sn(i) = sn(i) Xor sn(i - 2) Xor (k3 * sn(i - 1)) Mod 256: Next
For i = 1 To N - 1: sn(i) = sn(i) Xor sn(i + 1) Xor (k2 * sn(i + 1)) Mod 256: Next
For i = N To 2 Step -1: sn(i) = sn(i) Xor sn(i - 1) Xor (k1 * sn(i - 1)) Mod 256: Next

For i = 1 To N: Mid(ss, i, 1) = Chr(sn(i)): Next i

If salt Then StrDecode = Mid(ss, 3, Len(ss) - 4) Else StrDecode = ss

End Function

Public Function RES(resID As Integer) As String
    RES = LoadResString(g_LANGOFFSET + resID)
End Function

```